THESIS TF 142310

OPTIMASI KONDISI KOLOM DESTILASI BINER UNTUK MENCAPAI KUALITAS PRODUK DENGAN MENGGUNAKAN *IMPERIALIST COMPETITIVE ALGORITHM* (ICA)

NUR FITRIYANI
NRP 2414 201 010

Dosen Pembimbing :
Totok Ruki Biyanto, ST, MT, PhD

PROGRAM MAGISTER
BIDANG KEAHLIAN REKAYASA INSTRUMENTASI INDUSTRI
JURUSAN TEKNIK FISIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

THESIS TF 142310

OPERATIONAL OPTIMIZATION OF BINARY DISTILLATION COLUMN TO ACHIEVE PRODUCT QUALITY USING IMPERIALIST COMPETITIVE ALGPRITHM

NUR FITRIYANI
NRP 2414 201 010

Supervisor :
Totok Ruki Biyanto, ST, MT, PhD

MAGISTER PROGRAM
ENGINEERING OF INDUSTRIAL INSTRUMENTATION FIELD
DEPARTMENT OF ENGINEERING PHYSICS
FACULTY OF INDUSTRIAL TECHNOLOGY
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2016

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar

Magister Teknik (M.T)

di

Institut Teknologi Sepuluh Nopember

Oleh

NUR FITRIYANI

NRP. 2414 201 010

Tanggal Ujian : 28 Juli 2016

Periode Wisuda : September 2016

Disetujui oleh :

1. **Totok Ruki Biyanto, S.T., M.T., Ph.D.** ..... **(Pembimbing I)**
   NIP : 19710702 199802 1 001
2. **Dr. Ir. Ali Musyafa', M.Sc.** ..... **(Ketua Penguji)**
   NIP : 19600901 198701 1 001
3. **Dr. Ridho Hantoro, S.T., M.T.** ..... **(Penguji)**
   NIP : 19761223 200501 1 001
4. **Dr. Gunawan Nugroho, S.T., M.T.** ..... **(Penguji)**
   NIP : 19771127 200212 1 002

Direktur Program Pascasarjana

Prof. Ir. Djauhar Manfaat, M.Sc, Ph.D

NIP: 19601202 198701 1 001

OPTIMASI KONDISI OPERASI KOLOM DESTILASI BINER UNTUK MENCAPAI KUALITAS PRODUK DENGAN MENGGUNAKAN *IMPERIALIST COMPETITIVE ALGORITHM* (ICA)

Nama Mahasiswa	: Nur Fitriyani
NRP	: 2414201010
Jurusan	: Teknik Fisika-FTI ITS
Pembimbing	: Totok Ruki Biyanto, S.T., M.T., P.hD.

ABSTRAK

Kualitas produk dalam proses kimia harus dikontrol dan dioptimasi untuk mencapai kualitas terbaik dan mampu meminimalkan konsumsi energi. Kualitas produk pada proses kolom distilasi yang dijalankan dengan menggunakan teknik optimasi tidaklah mudah karena kolom distilasi bersifat non-linier. Kolom distilasi termasuk kedalam kelas NLP yang dapat dipecahkan dengan menggunakan metode stokastik untuk mencapai solusi global. Salah satu metode stokastik adalah ICA. ICA memiliki beberapa kelebihan yang dapat menyelesaikan permasalahan yang kompleks, dapat mencapai nilai konvergen yang lebih cepat dibandingkan dengan algoritma yang lain. Dalam penelitian ini, kolom distilasi biner dibangun dengan menggunakan Jaringan Syaraf Tiruan (JST) dengan input berupa molar feed, fraksi feed, molar *reflux* dan panas reboiler dengan output berupa *top product* dan *bottom product*. Selanjutnya ICA dijalankan untuk mengoptimasi error antara nilai dari prediksi JST dengan nilai *setpoint* yang diinginkan. Dari hasil optimasi pada penelitian ini, diperoleh setpoint pada laju aliran *reflux* harus sebesar 44.4169 kgmol/hr dan *steam* reboiler 9.0006e+005 kJ/hr untuk mendapatkan fraksi ethanol 0.9891 pada *top product* dan 0.007 fraksi ethanol pada *bottom product*.

Kata Kunci: Distilasi biner, Jaringan Syaraf Tiruan, ICA, Optimasi.

Halaman ini sengaja dikosongkan

OPERATIONAL OPTIMIZATION OF BINARY DISTILLATION COLUMN TO ACHIEVE PRODUCT QUALITY USING IMPERIALIST COMPETITIVE ALGORITHM (ICA)

Name	: Nur Fitriyani
NRP	: 2414201010
Department	: Engineering physics
Supervisor	: Totok Ruki Biyanto, S.T., M.T., P.hD.

ABSTRACT

The quality of a product in the chemical process need to controlled and optimized to achieve the best quality and minimize energy consumption. The quality in the process of the distillation column that performed by using optimization techniques is not easy because distillation column is nonlinear. The distillation column is an NLP class (Non-Linear Programming) which can be solved by using a stochastic method to achieve the global solution. ICA has several advantages that can solve the complex problems, increasing the speed of converging value than other algorithms. In this research, the binary distillation column model was built using a neural network (ANN) by changing the operational conditions of the column, i.e. molar flow feed, a composition of feed molar flow reflux, and reboiler heat duty to get the appropriate output i.e top product and bottom product. Then ICA was executed to optimize error between the predicted values of ANN with the desired setpoint value. The result of this study obtained setpoint of reflux rate must be at 44.4169 kg mol/hr and steam reboiler at 9.0006e+005 kJ/hr to obtain ethanol fraction 0.9891 of top product and 0.007 ethanol fraction of the bottom product.

Keywords : Binary distillation column; Optimization; Imperialist Competitive Algorithm.

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT, karena rahmat dan hidayah-Nya penulis diberikan kesehatan, kemudahan dan kelancaran penyusunan laporan thesis berjudul:

"OPTIMASI KONDISI KOLOM DISTILASI BINER UNTUK MENCAPAI KUALITAS PRODUK DAN DENGAN MENGGUNAKAN *IMPERIALIST COMPETITIVE ALGORITHM* (ICA)"

Thesis ini merupakan salah satu persyaratan akademik yang harus dipenuhi dalam Program Studi S-2 Teknik Fisika FTI ITS. Penulis menyempaikan terima kasih sebesar-besarnya kepada :

1. Direktorat Jenderal Pendidikan Tinggi, Departemen Pendidikan dan Kebudayaan Republik Indonesia, yang telah memberikan dukungan finansial kepada penulis melalui Beasiswa BPP-DN Program Fresh Graduate 2014-2016.
2. Totok Ruki Biyanto, S.T., M.T., Ph.D., selaku dosen pembimbing yang telah mengarahkan dan memberikan semangat penuh sehingga laporan ini dapat diselesaikan dengan baik.
3. Ibu dan Kakak yang selalu memberikan dukungan baik secara material maupun spiritual kepada penulis.
4. Dr. Ir. Ali Musyafa', M.Sc., Dr. Ridho Hantoro, S.T., M.T., dan Dr. Gunawan Nugroho, S.T., M.T., Selaku dosen penguji yang telah memberikan saran dan perbaikan paper dan thesis ini.
5. Agus Muhamad Hatta, S.T., M.Si., Ph.D., Selaku ketua jurusan Teknik Fisika ITS.
6. Dr. rer. Nat. Ir. Aulia M. T. Nasution M.Sc., Selaku ketua program studi S2 Teknik Fisika ITS.
7. Dr. Ing. Doty Dewi Risanti, S.T., M.T., Selaku dosen wali penulis.
8. Segenap Bapak/Ibu dosen pengajar di jurusan Teknik Fisika-ITS.
9. Rekan-rekan S2 Teknik Fisika ITS yang senantiasa menemani dan memberikan motivasi kepada penulis.

Penulis menyadari bahwa penulisan ini masih memiliki kekurangan, sehingga kritik dan dan saran yang membangun penulis harapkan dari rekan-rekan pembaca sekalian. Semoga laporan thesis ini dapat berguna dan bermanfaat bagi penulis dan pembaca. Aamiin.

Surabaya, 8 Agustus 2016

Penulis

DAFTAR ISI

ABSTRAK …….. iii

ABSTRACT ……... v

KATA PENGANTAR …… vii

DAFTAR ISI …….. ix

DAFTAR NOTASI ……. xi

DAFTAR GAMBAR …….. xix

DAFTAR TABEL …….. xxi

BAB I PENDAHULUAN …….. 1

1.1 Latar Belakang ……. 1

1.2 Perumusan Masalah ……..…… 4

1.3 Tujuan Penelitian ……. 5

1. 4 Manfaat Penelitian …….. 5

1.5 Lingkup Penelitian …… 5

1.6 Sistematika Laporan ……. 6

BAB II KAJIAN PUSTAKA DAN TINJAUAN PUSTAKA …… 7

2.1 Kajian Pustaka ……... 7

2.2 Tinjauan Pustaka ……... 8

2.2.1 Kolom Distilasi …… 8

2.2.2 Kesetimbangan uap cair ……... 10

2.2.3 Model Matematika …… 11

2.2.4 Jaringan Syaraf Tiruan ……. 15

2.2.5 *Imperialist Competitive Algorithm* (ICA) …… 18

BAB III METODOLOGI PENELITIAN …..……. 25

3.1 Flowchart Metodologi Penelitian ……..…… 25

3.2 Tahap-Tahap Penelitian ……..…… 25

BAB IV HASIL DAN PEMBAHASAN …… 33

4.1 Permodelan dinamik kolom distilasi biner ….…… 33

4.2 Perancangan Jaringan Syaraf Tiruan (JST) …… 39

4.3 Optimasi dengan menggunakan ICA …..…… 42

BAB V KESIMPULAN……..……... 47

DAFTAR PUSTAKA ……..……. 49

LAMPIRAN……..……. 53

DAFTAR NOTASI

$\lambda^{(0)}$.	Nilai awal
β	Koefisien sudut asimilasi
ξ	Zeta
B	*Bottom* produk (kgmole/hr)
C_n	Biaya normalisasi
c_n	Biaya dari banyaknya n
D	Distilat (kgmole/hr)
D	Vektor D
D_1	Vektor D 1
D_2	Vektor D 2
D_3	Vektor D 3
$D_{N_{imp}}$	Vektor jumlah *imperialist*
d	Jarak antara koloni terhadap imperialis
F	Umpan (*Feed*) (kgmole/hr)
$f^{(i)}$	Fungsi aktifasi
G	matrix baru
h_D	Entalpi *liquid* distilat (J/kg)
H_{NT}	Entalpi *vapour tray number* (J/kg)
h_{NT+1}	Entalpi *liquid tray number* keluaran (J/kg)
h_n	Entalpi *liquid* tiap tray masukan (J/kg)

h_{n+1}	Entalpi *liquid* tiap tray keluaran (J/kg)
H_n	Entalpi *vapour* tiap tray masukan (J/kg)
H_{n-1}	Entalpi *vapour* tiap tray keluaran (J/kg)
h_{NF+1}	Entalpi *liquid* tray umpan (J/kg)
h_{NF}	Entalpi *liquid* tray umpan (J/kg)
H_{NF-1}	Entalpi *vapour* tray umpan (J/kg)
h_F	Entalpi *liquid* feed (J/kg)
h_1	Entalpi *liquid* L_1(J/kg)
H_B	Entalpi *vapour bottom* produk (J/kg)
h_B	Entalpi *liquid bottom* produk (J/kg)
I	Iterasi
L_n	Liquid tray masukan
L_{n+1}	Liquid tray keluaran
L_{NT+1}	*Liquid tray number* keluaran
L_{NF}	Liquid tray umpan masukan
L_{NF+1}	Liquid tray umpan keluaran
L_1	Liquid bottom column keluaran
M_D	Neraca massa total distilat
$M_D X_D$	Neraca massa komponen distilat
$M_D h_D$	Neraca massa panas distilat
M_N	Neraca massa total tiap tray

$M_n X_n$	Neraca massa komponen tiap tray
$M_n h_n$	Neraca massa panas tiap tray
M_{NF}	Neraca massa total tray umpan
$M_{NF} X_{NF}$	Neraca massa komponen tray umpan
$M_{NF} h_{NF}$	Neraca massa panas tray umpan
M_B	Neraca massa total *bottom* produk
$M_B X_B$	Neraca massa komponen *bottom* produk
$M_B h_B$	Neraca massa panas *bottom* produk
n	Jumlah variabel atau banyaknya sample
N_{pop}	Jumlah populasi
N_{imp}	Jumlah imperialist
N_{col}	Jumlah koloni
$N.C_{.n}$	Inisialisasi jumlah koloni dari n empire
p_1	Variabel 1
p_2	Variabel 2
p_3	Variabel 3
$p_{N_{var}}$	Varabel jumlah n
P_{p_1}	Peluang Kepemilikan 1
P_{p_2}	Peluang Kepemilikan 2
P_{p_3}	Peluang Kepemilikan 3
$P_{pN_{imp}}$	Peluang kepemilikan tiap-tiap empire

P	Vektor P
P_n	Kekuatan normalisasi
Q_c	Panas reboiler (kJ/hr)
Q_R	Panas reboiler (kJ/hr)
r_1	Jumlah elemen 1
r_2	Jumlah elemen 2
r_3	Jumlah elemen 3
$r_{N_{imp}}$	Jumlah elemen empire
r	Harga awal
$r^{(i)}$	Harga baru
R	Jumlah elemen seragam
R	Vektor R
Y_{RB}	Komposisi reboiler
V_{NT}	Laju aliran *Vapour tray number* (kgmole/hr)
V_n	Laju aliran *Vapour* tiap tray keluaran (kgmole/hr)
V_{n-1}	Laju aliran *Vapour* tiap tray masukan (kgmole/hr)
V_{NF}	Laju aliran *Vapour* tray umpan keluaran (kgmole/hr)
V_{NF-1}	Laju aliran *Vapour* tray umpan masukan (kgmole/hr)
V_{RB}	Laju aliran *Vapour* reboiler (kgmole/hr)
v_1	Bobot sinyal 1
v_2	Bobot sinyal 2

V_N	Perkiraan error
w_1	Bobot sinyal 1
w_2	Bobot sinyal 2
w_3	Bobot sinyal 3
$w^{(0)}$	Bobot awal
$w^{(i)}$	Bobot baru
$w^{(i+1)}$	Bobot baru berikutnya
x	Jarak perpindahan koloni
X_D	Komposisi distilat
X_n	Komposisi tiap tray masukan
X_{n+1}	Komposisi tiap tray keluaran
X_{NF}	Komposisi tray umpan masukan
X_{NF+1}	Komposisi tray umpan keluaran
X_F	Komposisi feed
X_1	Komposisi liquid bottom
X_B	Komposisi *bottom* produk
X_1	Masukan neuron 1
X_2	Masukan neuron 2
X_3	Masukan neuron 3
x_1	Sinyal keluaran X_1
x_2	Sinyal keluaran X_2

x_3	Sinyal keluaran X_3
X_{exp}	Nilai eksperimental
X_{pre}	Nilai prediksi
Y	Keluaran neuron
y_{in}	Masukan jaringan
Y_{NT}	Komposisi *tray number*
Y_{NF-1}	Komposisi tray umpan keluaran
Y_n	Komposisi tray number keluaran
Y_{n-1}	Komposisi tray number masukan
Z_1	Unit keluaran 1
Z_2	Unit keluaran 2
Z^N	Pasangan data

DAFTAR SINGKATAN

ACO	*Ant Colony Optimization*
ANN	*Artificial Neural Network*
CDiC	*Conventional Distillation Column*
GA	*Genetic Algorithm*
HDIc	*Heat-Integrated Distillation Column*
HN	*Hidden Node*
ICA	*Imperialist Competitive Algorithm*
L-V	*Liquid Vapour*
LVC	*Less Component Volatile*
MCV	*More Component Volatile*
MIMO	*Multivariable Input Multivariable Output*
MLP	*Multy Layer Perceptron*
NARX	*Non-Linier Auto Regresive with External Input*
NLP	*Non Linear Programming*
PSO	*Particle Swarm Optimization*
PFD	*Process Flow Diagram*
P&ID	*Piping & Instrumentation Diagram*
RMSE	*Root Mean Square Error*
SPSA	*Simultaneous Perturbation Stochastic Approximation*
TS	*Tabu Search*

TAC	*Total Annual Cost*
VRC	*Vapour Compression Column*

DAFTAR GAMBAR

Gambar 2. 1 Kolom distilasi ……..…….……. 9

Gambar 2. 2 Kesetimbangan uap cair …….…… 10

Gambar 2. 3 Komposisi uap dan dan cairan pada kesetimbangan …….. 11

Gambar 2. 5 Kesetimbangan massa pada kondenser dan reflux drum …… 12

Gambar 2. 6 Kesetimbangan massa tiap tray ……... 13

Gambar 2. 7 Kesetimbangan massa pada tray umpan…….. 13

Gambar 2. 8 Kesetimbangan massa pada reboiler dan base kolom …… 14

Gambar 2. 9 Jaringan syaraf tiruan single-layer ……. 15

Gambar 2. 10 Jaringan syaraf tiruan multi-layer …… 16

Gambar 2. 11 Membangkitkan inisialisasi empire ……. 19

Gambar 2. 12 Pergerakan koloni menuju imperialis …….. 20

Gambar 2. 13 Perubahan posisi koloni dan imperialis, (b). Keseluruhan empire setelah perubahan posisi ……..…….…… 21

Gambar 2. 14 Kompetisi Imperialis …… 22

Gambar 3. 1 Diagram Alur Penelitian …….…… 25

Gambar 3. 2 Model sistem kolom distilasi biner …… 26

Gambar 3. 3 Model sistem kolom distilasi biner …… 27

Gambar 3. 4 Arsitektur Jaringan dari Plant …… 30

Gambar 3. 5 Flow Chart ICA…….…….. 31

Gambar 4. 1 Permodelan kolom distilasi …… 34

Gambar 4. 2 Profil pengaruh molar reflux terhadap komposisi distilat…..... 35

Gambar 4. 3 Profil pengaruh molar reflux terhadap komposisi bottom produk 36

Gambar 4. 4 Profil pengaruh panas reboiler terhadap komposisi distilat…….. 36

Gambar 4. 5 Profil pengaruh panas reboiler terhadap komposisi bottom produk 37

Gambar 4. 6 Profil pengaruh molar flow feed terhadap komposisi distilat…… 37

Gambar 4. 7 Profil pengaruh feed terhadap komposisi bottom produk……. 38

Gambar 4. 8 Profil pengaruh fraksi mol feed terhadap fraksi mol distilat …… 38

Gambar 4. 9 Profil pengaruh fraksi mol feed terhadap fraksi mol distilat …… 39

Gambar 4. 10 HN terhadap mean RMSE total …….. 40

Gambar 4. 11 Hasil pelatihan JST untuk prediksi fraksi mol distilat …… 41

Gambar 4. 12 Hasil pelatihan JST untuk prediksi fraksi mol bottom produk ... 41

Gambar 4. 13 Hasil validasi JST untuk prediksi fraksi mol distilat …….. 42

Gambar 4. 14 Hasil validasi JST untuk prediksi fraksi mol bottom produk …. 42

Gambar 4. 15 Proses iterasi dengan ICA ……..…….. 44

DAFTAR TABEL

Tabel 1. Spesifikasi sistem kolom distilasi 28

Tabel 2. Input dan output JST 29

Tabel 3. Parameter algoritma ICA 43

Tabel 4. Perbandingan hasil optimasi 44

Tabel 5. Hasil fitness error 44

Tabel 6. Hasil cost function saat parameter Country dirubah 45

Tabel 7. Hasil cost function saat parameter Imperialist dirubah 45

Tabel 8. Hasil cost function saat parameter Decade dirubah 45

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Bertambahnya permintaan energi dan adanya dampak negatif dari bahan bakar fosil terhadap lingkungan menuntut adanya pencarian sumber energi alternatif yang dapat diperbaharui salah satunya menggunakan alkohol sebagai *biofuels* seperti ethanol dan propanol (Pla-Franco, 2015). Parameter *biofuels* yang baik dapat dilihat dari tingkat kemurnian komposisi produk. Untuk mendapatkan kemurnian komposisi produk diperlukan proses pemisahan komponen salah satunya dengan proses distilasi.

Distilasi merupakan proses pemisahan dan pemurnian fraksi campuran yang berbeda titik didihnya. Proses distilasi meliputi penguapan cairan dengan cara pemanasan dilanjutkan dengan kondensasi uap menjadi cairan yang disebut sebagai *distillate*. Dalam prakteknya pemilihan prosedur distilasi bergantung pada sifat cairan yang akan dimurnikan dan sifat pengotor yang terkandung didalamnya. Ethanol dapat diperoleh dengan berbagai cara yaitu hidrasi etilen, fermentasi glukosa dan hasil samping kegiatan industri (*side stream*). Proses dehidrasi ethanol tidak dapat dilakukan dengan mudah karena ethanol dan air adalah azeotrop yang tidak dapat dilakukan pada kolom distilasi biasa sehingga memerlukan teknik lain seperti distilasi ekstraktif. Distilasi ekstraktif merupakan distilasi dengan penambahan entrainer yang digunakan untuk memecah azeotrop dan memudahkan proses pemisahan (Pla-Franco dkk., 2015)

Distilasi banyak dipakai luas dalam industri petrokimia, *refinery*, industri farmasi dan industri makanan. Pada tahun 1995, Humprey, mengestimasi sekitar 90% distilasi ditangani oleh teknik pemisahan dan pemurnian (Hosgor, dkk., 2014). Prinsip dari teknik distilasi adalah pemisahan berdasarkan perbedaan titik didih untuk mendapatkan komposisi destilat dan produk bawah yang sesuai (Luyben, 1997). Komposisi produk yang sesuai dipengaruhi oleh beberapa variabel diantaranya besarnya *input* panas pada *reboiler*, rasio *reflux* pada bagian atas kolom, komposisi dan laju aliran umpan (*feed flow rate*).

Proses distilasi memerlukan konsumsi energi yang tinggi, yaitu sekitar 50% dari biaya operasional proses sehingga diperlukan beberapa teknik penghematan konsumsi energi diantaranya adalah teknik injeksi uap *superheated* (Samborskaya, 2014), teknik integrasi panas (Shahandeh, 2015), teknik pengendalian *thermal* atau adiabatis kolom distilasi (Schaller, 2001), dan metode penentuan konfigurasi kolom distilasi untuk minimumkan energi pada kolom tunggal (Lucia & McCallum, 2010).

Cara lain dalam menentukan penghematan energi adalah dengan melakukan teknik optimisasi. Teknik optimisasi merupakan sebuah tendensi dalam menentukan solusi terbaik dalam batasan-batasan (*constrains*) yang telah ditentukan. (Caballero & Grossmann, 2014) dalam papernya mengoptimisasi kolom distilasi meliputi pemilihan nomor tray, penempatan lokasi umpan, dan kondisi operasi dalam meminimalisir *TAC (Total Annual Cost)* atau total biaya tahunan termasuk biaya modal dan biaya operasional. Tantangan utama dalam mendesain kolom distilasi adalah menentukan dimensi yang tepat pada kolom seperti diameter, ketinggian dan kondisi operasionalnya sehingga komposisi produk dapat dicapai dengan biaya modal dan biaya operasional yang minimal (Sorensen, 2014a). Penelitian lain mengenai optimisasi dilakukan untuk mencari kondisi operasi pada *reflux* rasio dan panas *reboiler* dengan tujuan untuk mendapatkan energi minimum (Biyanto dkk., 2015). Komposisi dan kebutuhan energi pada kolom distilasi juga dipengaruhi oleh tekanan pada top kolom distilasi sehingga hal ini juga perlu dipertimbangkan untuk dilakukan optimasi (Sorensen, 2014b).

Optimasi pada kolom distilasi adalah non-linier yang mungkin mempunyai beberapa lokal optimum sehingga optimasi kolom distilasi digolongkan kedalam kelas NLP (*Non-Linier Programming*) (Sorensen, 2014b). Metode NLP sangatlah kompleks sehingga memerlukan beberapa penyederhanaan(Ramanathan dkk., 2001). NLP dapat dipecahkan dengan dua cara, yaitu dengan pendekatan algoritma determinisktik dan stokastik. Algoritma deterministik menghasilkan solusi optimal dan memerlukan pemahaman proses dengan mengasumsikan konveksitas untuk mencapai konvergen dan solusi global optimum, ketika diaplikasikan ke dalam permasalahan non-konvek algoritma ini kemungkinan tidak dapat mencapai global optimum. Sedangkan Algoritma stokastik didasarkan pada metode adaptif pencarian acak, metode tersebut

tidak memerlukan langkah identifikasi struktur masalah dan mampu mengeliminasi sumber non-konvek sehingga dapat mencapai konvergen dan menjamin global optimum, selain itu algoritma stokastik memiliki kemampuan perhitungan yang intensive dalam menemukan pencarian acak dengan jangkauan yang luas pada aplikasi teknik dan kecepatan teknologi perhitungan modern (Ramanathan dkk., 2001), oleh karena itu algoritma stokastik digunakan sebagai pilihan terbaik untuk menyelesaikan masalah non-konvek dan dapat mengoptimasi problem yang kompleks (Scwhefel, 1993).

Pada sebagian besar metode algoritma stokastik seperti GA (*Genetic Algorithm*), TS (*Tabu Search*) merupakan tipe yang lamban dalam menentukan solusi optimal sedangkan pada beberapa metode stokastik baru seperti PSO (*Particle Swarm Optimization*), ACO (*Ant Colony Optimization*) dan ICA (*Imperialist Competitive Algorithm*), memiliki beberapa keunggulan diantaranya dapat menyelesaikan permasalahan yang sulit dan kompleks, tidak hanya memberikan respon yang lebih baik tetapi juga lebih cepat mencapai konvergen dibandingkan algoritma evolusi biasa (Niknam dkk., 2011). Pada penelitian yang dilakukan oleh Ramanathan, optimasi pada kolom distilasi kontinu dengan menggunakan dua pendekatan stokastik yaitu GA dan SPSA (*Simultaneous Perturbation Stochastic Approximation*) dihasilkan bahwa kedua metode tersebut dapat meminimalisir biaya total distilasi. Penelitian ini dilakukan dengan cara memanipulasi variabel berupa jumlah stage dan penentuan lokasi umpan sehingga proses optimasi dijalankan dengan menitikberatkan desain konfigurasi kolom belum sampai pada kondisi operasional kolom. Pada penelitian lain, algoritma ICA diterapkan pada *design heat exchanger* didapatkan nilai penurunan yang lebih rendah dibandingkan GA. Hasil penelitian tersebut menunjukan nilai penurunan ICA yang lebih besar daripada GA.

ICA diusulkan oleh Aztahpaz Gargary dan Lucas pada tahun 2007. ICA adalah teknik yang menjanjikan dalam menyelesaikan masalah NLP. Solusi NLP menggunakan ICA pada penelitian sebelumnya terbukti menjadi pendekatan yang valid untuk masalah non-konvek dimana waktu komputasi selalu minimal sehingga *run-time* nya jauh lebih efisien (Biyanto, dkk., 2015). ICA merupakan salah satu metode evolusi algoritma terkuat yang telah digunakan secara *ekstensive* dalam

menyelesaikan permasalahan optimisasi yang berbeda (Atashpaz-Gargari & Lucas, 2007). Metode ini didasarkan pada proses *socio-politic* yang dimotivasi oleh strategi pencarian global berupa kompetisi antar imperialist. ICA dimulai dengan menginisialisasi populasi. Pada metode algoritma ini, tiap-tiap individu dari populasi disebut *country,* beberapa *best country* dalam populasi dipilih sebagai pusat imperialist dan country yang lain membentuk *colony* dalam imperial tersebut. Setelah membagi *colony* selanjutnya membentuk inisialisasi *empire* dan *colony* bergerak menuju ke imperial country yang relevan. Pergerakan *colony* ke *empire* ini membentuk kompetisi dan juga mekanisme *collapse* sehingga beberapa country menjadi konvergen dan hanya satu empire terpilih. Proses inilah yang menjadikan ICA sebagai teknik terkuat meskipun teknik ini dapat menyebabkan local optima khususnya ketika jumlah imperial ditambahkan (Atashpaz-Gargari & Lucas, 2007). ICA telah diaplikasikan dalam menentukan batas tepi *image* (Zaynab dkk., 2014), penyelesaian perjalanan *salesman* (Xu, Wang & Huang, 2014), memprediksi formasi *hydrate temperature* (Hadi, dkk., 2014), mendesain optimal controller (Abdechiri, 2008) dan juga dalam mengoptimisasi desain *green hybrid power system* (Gharavi, dkk., 2015).

Pada penelitian ini *Imperial Competitive Algorithm* (ICA) digunakan untuk mengoptimasi kondisi operasional kolom dengan cara merubah laju aliran *reflux* dan *steam* pada *reboiler* sehingga didapatan kualitas kadar ethanol yang sesuai serta penghematan energi dan diharapkan kolom distilasi dapat dioperasikan lebih efisien, baik dari segi kualitas produk maupun pemakaian energinya.

1.2 Perumusan Masalah

Berdasarkan pada latar belakang diatas maka dapat dirumuskan permasalahan sebagai berikut:

a. Bagaimana pengaruh laju aliran umpan (*molar flow feed*), komposisi umpan (*mole fraction of feed*), laju aliran *reflux* (*molar flow reflux*) dan panas pada reboiler (*heat duty of reboiler*) terhadap komposisi kemurnian produk?

b. Bagaimana hasil kuantitas dan kualitas produk yang optimal dan penghematan energi yang sesuai dengan menggunakan *imperialist competitive algorithm* (ICA)?

1.3 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah untuk

a. Menganalisis pengaruh laju aliran umpan (*molar flow feed*), komposisi umpan (*mole fraction of feed*), laju aliran *reflux* (*molar flow reflux*) dan panas pada reboiler (*heat duty of reboiler*) terhadap komposisi kemurnian produk.
b. Merancang optimasi sistem pada kolom distilasi biner *ethanol-1 propanol* agar diperoleh kualitas produk yang sesuai dengan menggunakan ICA (*imperialist competitive algorithm*).

1.4 Manfaat Penelitian

Manfaat dari penelitian adalah dapat mengembangkan ilmu dalam aplikasi study optimasi menggunakan ICA (*imperial competitive algorithm*) sehingga diperoleh komposisi produk yang sesuai dan hemat energi.

1.5 Lingkup Penelitian

Pada penelitian kali ini, lingkup penelitian dilakukan dengan mengidentifikasi data, selanjutnya melakukan permodelan kolom distilasi biner, melakukan perancangan jaringan syaraf tiruan, permodelan formulasi dan teknik optimasi dengan menggunakan ICA (*Imperial Competitive Algorithm*) secara paralel atau bersamaan, pada permodelan kolom distilasi selanjutnya dilakukan validasi dengan data PFD (*Process Flow Diagram*) jika sesuai maka bersamaan dengan permodelan formulasi dan teknik optimasi dilakukan analisis untuk selanjutnya ditarik kesimpulan.

1.6 Sistematika Laporan

Secara sistematis, laporan thesis ini tersusun dalam lima bab dengan penjelasan sebagai berikut:

BAB I Pendahuluan

Bab ini berisi latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, lingkup penelitian dan sistematika laporan thesis.

BAB II Tinjauan Pustaka

Bab ini berisi tentang kajian pustaka dari penelitian-penelitian sebelumnya serta tinjauan pustaka mengenai teori-teori yang mendasari penelitian. Teori-teori tersebut meliputi permodelan kolom distilasi biner dengan struktur pengendalian L-V (*Liquid-Vapor*), Perancangan jaringan syaraf tiruan hingga penerapan *imperialist competitive algorithm*.

BAB III Metodologi Penelitian

Pada bab ini dijelaskan langkah-langkah penelitian dimulai dari permodelan kolom distilasi biner dengan struktur pengendalian L-V (*Liquid-Vapor*), Perancangan jaringan syaraf tiruan hingga penerapan *imperialist competitive algorithm.*

BAB IV Analisa Data dan Pembahasan

Pada bab ini berisi data serta analisis dan pembahasan yang didapatkan dari proses pengolahan data.

BAB V Kesimpulan dan Saran

Pada bab akhir ini memaparkan kesimpulan dan saran yang terkait dengan penelitian yang telah dilaksanakan agar dapat dilakukan pengembangan-pengembangan penilitian berikutnya.

BAB II
KAJIAN DAN TINJAUAN PUSTAKA

2.1 Kajian Pustaka

Penelitian ini menitikberatkan pada studi optimasi kolom distilasi *biner*. Dimana teknik optimasi memiliki tujuan untuk menemukan solusi optimal yang berkaitan dengan masalah variabel guna mencapai standar yang diinginkan. Banyak teknik yang digunakan dalam mengoptimalkan suatu sistem, diantaranya dengan menggunakan *evolution* algoritma.

Sorensen pada tahun 2014 dalam penelitiannya menyebutkan bahwa dengan menganalisis parameter sensitivitas (berupa laju aliran uap panas *superheated*, formasi air-hidrokarbon azeotrop, beban fluktuasi panas pada tray kolom) pada kolom distilasi dan heat exchanger dengan menggunakan ANN (*Artificial Neural Network*) dimungkinkan untuk dapat menghemat energi sebesar 21% (Sorensen, 2014b).

Dari penelitian lain oleh Shahadeh pada tahun 2014 menyebutkan bahwa dengan mengintegrasi panas ke dalam kolom distilasi dapat memperbaiki efisiensi thermodinamic yang rendah dan konsumsi energi yang besar. Metode integrasi panas dilakukan dengan mengganti *heat pump* CDiC (*Conventional distillation column*) dengan VRC (*Vapour compression column*) dan HIDc (*Heat-Integrated Distillation column*) kemudian mengintegrasikannya. Cara ini tidak dapat mengurangi reduksi TAC (*Total Annual Cost*) secara signifikan (13%) karena biaya modal relatif besar (Shahandeh dkk., 2015).

Pada penelitian oleh Ramanathan pada tahun 2001 mengoptimasi kolom ditilasi kontinu pada pemisahan alcohol dan air dengan menggunakan metode stokastik (SPSA dan GA) dengan variable yang dimanipulasi adalah jumlah stage, rasio *reflux* dan pemilihan lokasi umpan sehingga didapatkan kualitas produk yang sesuai. Hasilnya SPSA dan GA dapat meningkatkan hasil dan kinerja kolom distilasi kontinu.

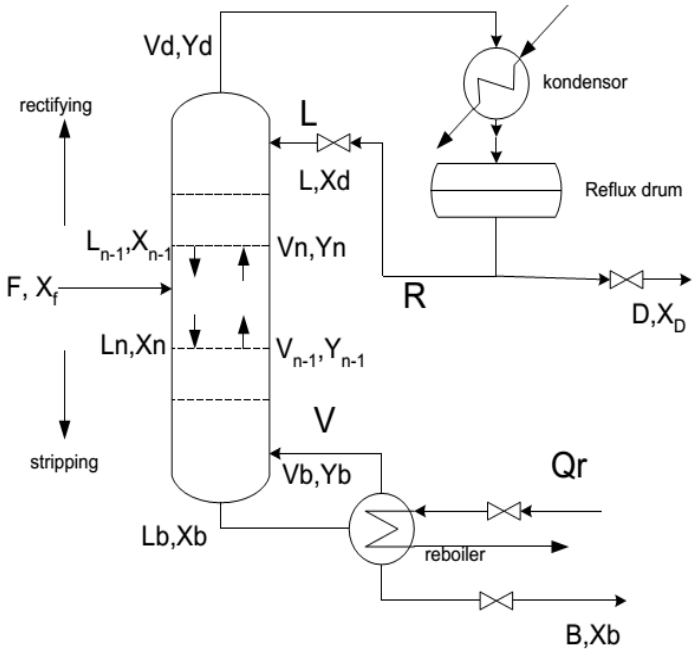
2. 2 Tinjauan Pustaka

2.2.1 Kolom Distilasi

Kolom distilasi digunakan sebagai unit pemisahan campuran dalam industri kimia. Metode distilasi didasarkan pada pemanasan campuran untuk menghasilkan dua fase (fase uap dan fase liquid). Operasi kolom distilasi biner diawali ketika feed masuk kedalam stage kolom yang membagi dua bagian yaitu *stripper* dan *rectifiyer*. Uap yang dihasilkan oleh reboiler terdapat pada bagian bawah kolom. Selanjutnya uap bergerak keatas kolom karena memiliki nilai volatile yang tinggi atau *More Volatile Component* (MCV), dan selanjutnya dikondensasikan oleh kondenser, Jumlah kondensat selanjutnya dikumpulkan oleh *reflux drum* untuk selanjutnya diumpankan kembali kedalam kolom agar dapat mencapai kemurnian yang diinginkan. *Reflux* bergerak kebawah dan disebut sebagai *Less Volatile Component(LVC)*. (Ramanathan dkk., 2001).

Distilasi digunakan pada hampir 90-95% industri pemisahan. Sebagian besar pemanasan campuran akan menghasilkan fase uap yang memiliki komponen volatile lebih banyak dibandingkan dengan fase liquid. Tantangan utama dalam mendesain distilasi kolom adalah menentukan dimensi kolom dengan benar seperti tinggi dan diameter kolom, serta operasi kondisi yang sesuai yang umumnya ditentukan oleh input panas pada reboiler dan jumlah rasio *reflux* yang dihasilkan pada atas kolom sehingga kemurnian produk dapat dicapai dengan biaya modal dan biaya operasional yang minimum (Sorensen, 2014b).

Pada penelitian ini kolom distilasi yang digunakan adalah kolom distilasi biner dengan komponen campuran berupa ethanol (C_2H_5OH) dan 1-propanol ($CH_3CH_2CH_2OH$). Proses pemisahan biner kedua komponen tersebut dapat dilakukan jika keduanya memiliki titik didih yang berbeda. Komponen ethanol memiliki titik didih sekitar 78, 29 ^oC dan 1-propanol memiliki titik didih sekitar 97^oC. Dari segi batas titik didihnya maka komponen ethanol akan menguap terlebih dahulu kemudian diikuti 1-propanol.



Gambar 2. 1 Kolom Distilasi (http://newcastle.edu.au)

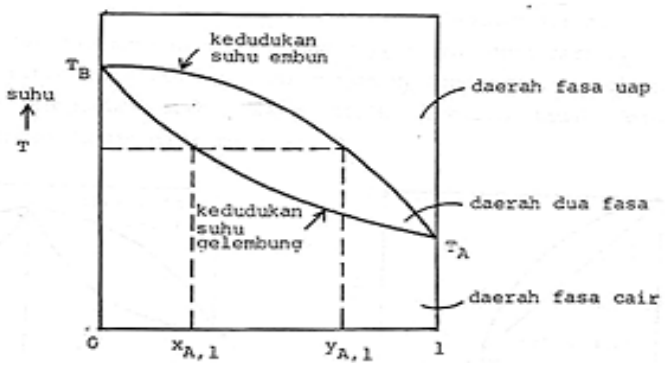
Bagian-bagian kolom distilasi :

Tray	:	Tempat terjadinya pertukaran panas cairan atas kolom menuju bawah, dengan fraksi uap dari bawah menuju atas; pertukaran panas ini untuk meningkatkan kemurnian fraksi yang diambil sebagai distilat.
Reboiler	:	Digunakan untuk memanaskan kembali umpan (feed) pada bagian dasar kolom distilasi, tipe lain reboiler adalah steam yang langsung masuk kedalam dasar kolom distilasi untuk memanaskan umpan.
Condenser	:	Mengkondensasikan overhead product (destilat) yang akan ditampung pada akumulator (reflux drum).
Reflux	:	Distilat yang dikembalikan kedalam kolom distilasi. Berfungsi untuk meningkatkan kemurnian distilat.

2.2.2 Kesetimbangan Uap-Cair (*Vapour Liquid Equilibrium*)

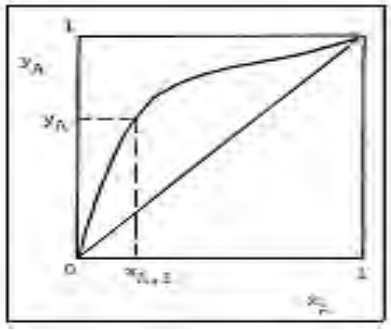
Perhitungan pada kolom distilasi berdasarkan pada data VLE (*Vapor-Liquid Equilibrium*). Keberhasilan suatu operasi distilasi tergantung pada keadaan setimbang yang terjadi antara fasa uap dan fasa cairan dari suatu campuran. Dalam hal ini akan ditinjau campuran biner yang terdiri dari kompoenen A (yang lebih mudah menguap) dan komponen B (yang kurang mudah mengaup). Karena pada umumnya proses distilasi dilaksanakan dalam keadaan dimana suatu campuran cairan mulai menguap (*buble temperature*) dan keadaan dimana suatu campuran gas mulai mengembun (*dew temperature*). Komposisi uap ditunjukkan pada Gambar 2.2 sedangkan komposisi uap dan cairan yang ada dalam kesetimbangan ditunjukkan pada Gambar 2.3.

Dalam banyak campuran biner, titik didih campuran terletak di antara titik didih komponen yang lebih mudah menguap (Ta) dan titik didih komponen yang kurang mudah menguap (Tb). Untuk setiap suhu, harga yA selalu lebih besar daripada harga xA. Ada beberapa campuran biner yang titik didihnya di atas atau di bawah titik didih kedua komponennya. campuran azeotrop minimum seperti pada Gambar 2.2. Dalam kedua hal, yA tidak selalu lebih besar daripada harga xA, ada kesetimbangan uap cairan dengan yA selalu lebih kecil daripada xA. Pada titik azeotrop, yA sama dengan xA dan campuran cairan dengan komposisi sama dengan titik azeotrop tidak dapat dipisahkan dengan cara distilasi keadaan ini disebut juga sebagai komposisi kritis (*critical composition*).

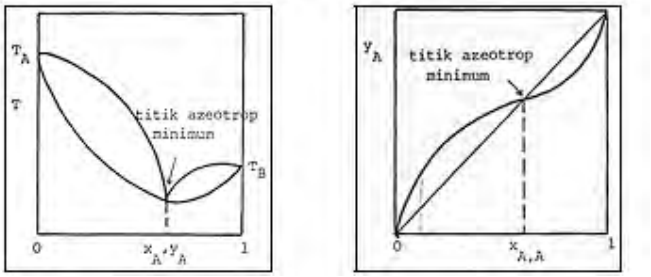


Gambar 2. 2 Kesetimbangan uap cair pada temperature *buble* dan temperatur *dew* (Sorensen, 2014a).

$X_{A,1}$ dan $y_{A,1}$ adalah komposisi cairan dan uap pada keadaan setimbang



Gambar 2. 3 Komposisi uap dan cairan pada kesetimbangan (Sorensen, 2014a).



Gambar 2. 4 Kurva azeotrop minimum pada kesetimbangan (Sorensen, 2014a).

2.2.3 Model Matematika

Kolom destilasi biner mempunyai kesetimbangan massa dan energi sehingga perhitungan kolom destilasi berdasarkan pada data kesetimbangan uap dan liquid. Kesetimbangan mass and energy balance dari kolom distilasi biner dijelaskan sebagai berikut.

Kesetimbangan total dari kolom distilasi:

$$F = D + B \tag{2. 1}$$

Kesetimbangan massa komponen kolom distilasi:

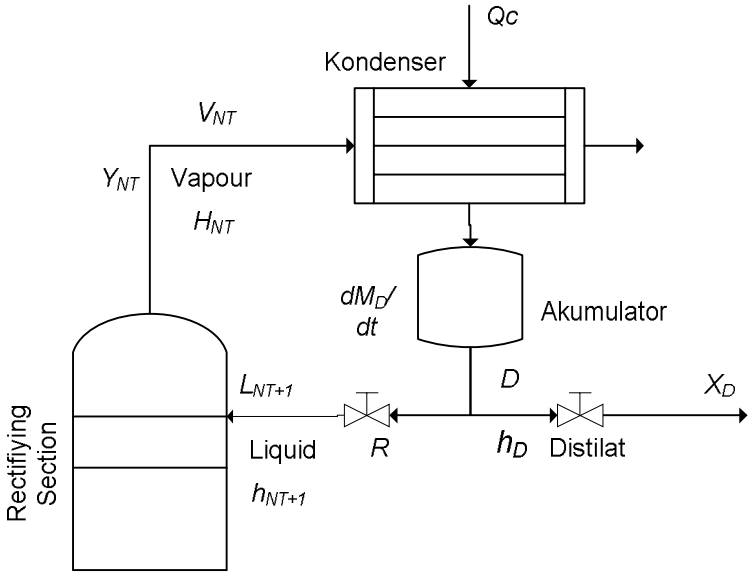
$$FX_f = DX_D + BX_B \tag{2. 2}$$

Kesetimbangan energi total:

$$Fh_F = Dh_D + Bh_B + Q_r - Q_c \tag{2. 3}$$

Adapun model matematis di tiap bagiannya dapat diterangkan sebagai berikut:

- Kesetimbangan massa pada kondenser dan *reflux* drum



Gambar 2. 5 Kesetimbangan massa pada kondenser dan *reflux* drum (Luyben, 1997).

Neraca massa total :

$$\frac{dM_D}{dt} = V_{NT} - L_{NT+1} - D \qquad (2.\ 4)$$

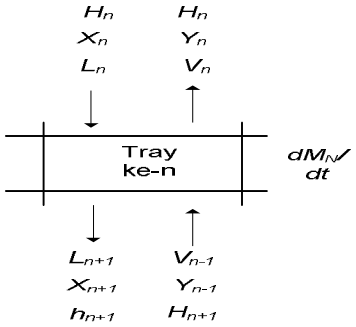
Neraca massa komponen

$$\frac{d(M_D X_D)}{dt} = V_{NT} Y_{NT} - (L_{NT+1} + D) X_D \qquad (2.\ 5)$$

Neraca panas

$$\frac{d(M_D h_D)}{dt} = V_{NT} H_{NT} - L_{NT+1} h_{NT+1} - D h_D - Q_c \qquad (2.\ 6)$$

- Kesetimbangan massa tiap tray



Gambar 2. 6 Kesetimbangan massa tiap tray (Luyben, 1997).

Neraca massa total

$$\frac{dM_n}{dt} = L_{n+1} - L_n - V_{n-1} + V_n \tag{2. 7}$$

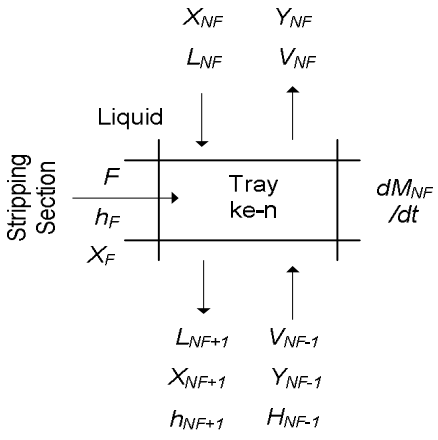
Neraca massa komponen

$$\frac{d(M_n X_n)}{dt} = L_{n+1}X_{n+1} - L_n X_n - V_{n-1}Y_{n-1} + V_n Y_n \tag{2. 8}$$

Neraca massa panas

$$\frac{d(M_n h_n)}{dt} = L_{n+1}h_{n+1} - L_n H_n - V_{n-1}H_{n-1} + V_n H_n \tag{2. 9}$$

- Kesetimbangan massa pada tray umpan (n=Np)



Gambar 2. 7 Kesetimbangan massa pada tray umpan

Neraca massa total

$$\frac{dM_{NF}}{dt} = L_{NT+1} - L_{NF} + F - V_{NF-1} + V_{NF} \quad (2.\ 10)$$

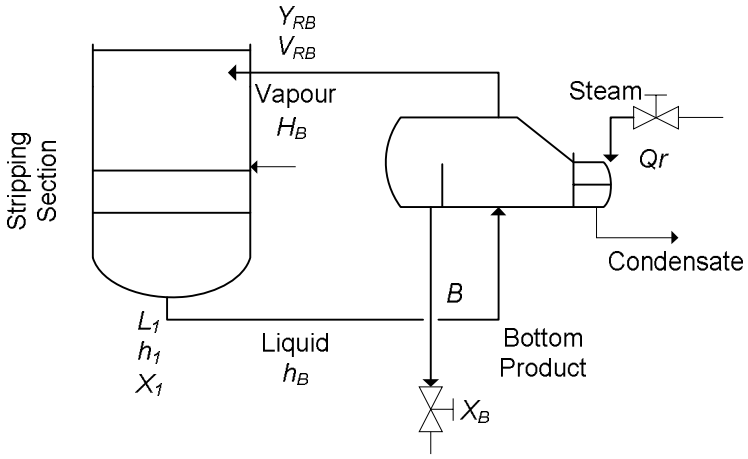
Neraca massa komponen

$$\frac{d(M_{NF}X_{NF})}{dt} = L_{NF+1}X_{NF+1} - L_{NF}X_{NF} - V_{NF-1}Y_{NF-1} + FX_F + V_{NF}Y_{NF} \quad (2.\ 11)$$

Neraca massa panas

$$\frac{d(M_{NF}h_{NF})}{dt} = L_{NF+1}h_{NF+1} - L_{NF}h_{NF} - V_{NF-1}H_{NF-1} + Fh_F + V_{NF}H_{NF} \quad (2.\ 12)$$

- Kesetimbangan massa pada reboiler dan base kolom



Gambar 2. 8 Kesetimbangan massa pada reboiler dan base kolom (Luyben, 1997).

Neraca massa total

$$\frac{dM_B}{dt} = L_1 - V_{RB} - B \quad (2.\ 13)$$

Neraca massa komponen

$$\frac{d(M_BX_B)}{dt} = L_1X_1 - V_{RB}Y_{RB} - BX_B \quad (2.\ 14)$$

Neraca panas

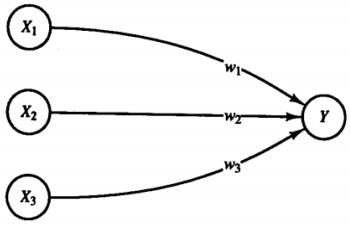
$$\frac{d(M_B h_B)}{dt} = L_1 h_1 - V_{RB} H_B - B h_B + Q_R \qquad (2.\ 15)$$

2.2.4 Jaringan Syaraf Tiruan

Jaringan syaraf tiruan atau JST dalam penelitian ini digunakan sebagai model pembentuk *plant*. JST didefinisikan sebagai suatu sistem pemrosesan informasi yang memiliki karakteristik menyerupai jaringan syaraf tiruan manusia. Jaringan syaraf tiruan merupakan salah satu model yang tidak bergantung pada asumsi model dan distribusi probabilitas (Demuth, 2002). Sistem jaringan syaraf tiruan juga disebut sebagai *brain methapor*, *computational neuroscience* atau *parallel distributed processing*.

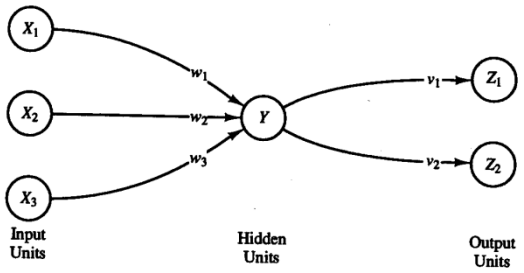
JST dikarakterisasi dengan percabangan koneksi antar neuron yang disebut sebagai arsitektur JST, metode ini menentukan bobot koneksi yang disebut sebagai training, learning dan algorithm serta fungsi aktifasi. JST terdiri dari elemen pemrosesan sederhana yang disebut sebagai neuron, units, sel dan node. Tiap-tiap neuron terkoneksi dengan neuron lain secara langsung dengan tiap neuron terhubung dengan bobot. Bobot jaringan direpresentasikan sebagai informasi yang akan digunakan oleh jaringan untuk menyelesaikan problem. Tiap-tiap neuron memiliki internal state yang disebut sebagai fungsi aktifasi atau fungsi yang diterima input. Gambar 2.9 merupakan model satu neuron Y, yang menerima input dari neuron X_1, X_2 dan X_3 secara berturut-turut. Aktivasi atau sinyal output dari dari ketiga neuron tersebut adalah x_1, x_2 dan x_3. Bobot yang menghubungkan X_1, X_2 dan X_3 yaitu w_1, w_2 dan w_3. Input jaringan, y_{in}, terhadap neuron Y merupakan penjumlahan bobot sinyal dari neuron X_1, X_2 dan X_3 yaitu :

$$y_{in} = w_1 x_1 + w_2 x_2 + w_3 x_3 \qquad (2.\ 16)$$



Gambar 2. 9 Jaringan syaraf tiruan single-layer

Untuk struktur jaringan multilayer, neuron terdiri dari banyak layer yaitu input layer X_1, X_2 dan X_3, output layer Z_1 dan Z_2 dan hidden layer Y yang terletak diantara input layer dan output layer. Jaringan multilayer dapat menyelesaikan permasalahan yang lebih rumit dibandingkan dengan single layer.



Gambar 2. 10 Jaringan syaraf tiruan multi-layer (Fausett, 2015)

Pemilihan struktur model digunakan untuk mengetahui struktur fisis dari sistem. Dari sudut pandang identifikasi sistem, jaringan syaraf tiruan memiliki beberapa kelebihan diantaranya dapat digunakan pada sistem non-linier, dapat diaplikasikan pada sistem MIMO proses (*multivariable input multivariable output*) dan juga memiliki kemampuan adaptasi dan training data. Jaringan syaraf tiruan merupakan modelling data yang mampu mengetahui secara langsung hubungan variabel masukan dan keluaran melalui iterasi pelatihan (Popova, Iliev, & Trifonov, 2011). Algoritma pada proses pelatihan tergantung pada kompleksitas masalah, jumlah data pada saat proses pelatihan, jumlah bobot dan bias jaringan dan error yang ingin dicapai (Demuth, 2002). JST memiliki beberapa keterbatasan diantaranya kekurangmampuannya dalam melakukan operasi-operasi numerik dangan presisi tinggi, operasi algoritma aritmatik, operasi logika, serta operasi simbolis serta lamanya proses pelatihan yang membutuhkan waktu yang berhari-hari untuk jumlah data yang besar.

Algoritma pelatihan yang digunakan dalam penelitian ini adalah algoritma Levenberg-Maquardt, algoritma ini jauh lebih kompleks jika dibandingkan dengan algoritma back-propagation. Dan berikut langkah-langkah algoritma Lavenberg-Maquardt:

1. Pilih bobot awal berupa vektor $w^{(0)}$ dan nilai awal $\lambda^{(0)}$. Dimana w adalah bobot dan λ diberikan harga awal.
2. Tentukan arah pencarian dari persamaan berikut :

$$Rw^{(i)} + \lambda^{(i)} I \; f^{(i)} = -Gw^{(i)} \tag{2.17}$$

Diperoleh f dan dimasukkan ke persamaan:

$$w = \arg\min_{w} V_N \; (w, Z^N) \tag{2.18}$$

Jika $V_N\left(w^{(i)} + f^{(i)}, Z^N\right) < V_N\left(w^{(i)}, Z^N\right)$ sehingga memenuhi $w^{(i+1)} = w^{(i)} + f^{(i)}$ sebagai iterasi baru maka $\lambda^{(i+1)} = \lambda^{(i)}$. Dan jika tidak maka mencari harga baru r sebagai berikut:

$$r^{(i)} = \frac{V_N w^{(i)}, Z^N - V_N w^{(i)} + f^{(i)}, Z^N}{V_N w^{(i)}, Z^N - L^{(i)} w^{(i)} + f^{(i)}} \tag{2.19}$$

Apabila $r^{(i)} > 0.75$ maka $\lambda^{(i)} = \lambda^{(i)} / 2$

Apabila $r^{(i)} < 0.75$ maka $\lambda^{(i)} = \lambda^{(i)} * 2$

3. Apabila kriteria tercapai, maka perhitungan berhenti, dan jika kriteria belum tercapai maka kembali mengulang ke langkah nomor 2.

(Norgaard, 2000)

Performansi model jaringan syaraf tiruan dianalisa dengan menggunakan *Root Mean Square Error* (RMSE), RMSE merupakan analisis error statistik yang digunakan untuk mengecek keakuratan model. RMSE digunakan untuk menentukan rata-rata kesalahan pada model yang telah dirancang. Jika nilai RMSE yang dihasilkan kecil, maka model yang telah dirancang semakin valid, dan sebaliknya. Korelasi keakuratan sebanding dengan nilai eksperimental yang ditentukan dengan teknik statistik yang berbeda. Nilai terbaik antara eksperimental dan prediksi tidak mungkin memiliki nilai RMSE sama dengan nol. Proses pelatihan dilakukan pada penelitian ini dengan menggunakan komposisi 80% data training dan 20% data untuk validasi (Makarynskyy, 2004). Data training dilakukan untuk melatih kemampuan model dan data validasi digunakan untuk memeriksa keakuratan model. Pemilihan komposisi

80% dan 20% untuk mendapatkan prediksi RMSE dan koefisien korelasi terbaik (Nitsure et al, 2012). RMSE didefinisikan sbb:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_{exp} - X_{pre})^2} \qquad (2.\ 20)$$

(Hadi et al., 2014).

2.2.5 Imperial Competition Algorithm (ICA)

ICA diklasifikasikan sebagai metode berdasarkan populasi seperti algoritma genetik. Metode ini merupakan salah satu bentuk *socio politic* yang dimotivasi oleh strategi pencarian global yang belakangan ini telah diperkenalkan untuk menentukan bentuk optimisasi yang berbeda. ICA dapat digunakan pada metode optimisasi karena *run-time* nya dapat diterima dan efektif. (Atashpaz-Gargari & Lucas, 2007) mempersembahkan ide evolutioner algoritma. Berikut langkah metode ICA :

a). Inisialisasi Empire

Tujuan utama dalam optimisasi adalah untuk menemukan solusi optimal yang berkaitan dengan masalah variabel. Atashpaz dan GE Lucas membentuk susunan nilai variabel agar dapat dioptimisasi. Dalam Algoritma Genetik, susunan ini disebut sebagai ‘kromosom’ akan tetapi dalam ICA disebut sebagai ‘country’. Susunannya didefinisikan sebagai berikut:

$$country = [p_1, p_2, p_3, \dots, p_{N_{var}}] \qquad (2.\ 21)$$

Nilai variable pada country digambarkan sebagai persebaran jumlah titik. Harga dari country dapat diperoleh dengan mengevaluasi fungsi biaya f pada variabel $[p_1, p_2, p_3, \dots, p_{N_{var}}]$, sehingga persamaan menjadi:

$$cost = f(country) = f(p_1, p_2, p_3, \dots, p_{N_{var}}) \qquad (2.\ 22)$$

Langkah awal optimisasi algoritma yaitu proses inisialisasi ukuran populasi N_{pop}. Dipilih N_{imp} sebagai *country* terkuat dalam suatu *empire*. Dan sisanya populasi N_{col} menjadi koloni pada tiap-tiap *empire*. Sehingga diperoleh dua tipe *country* yaitu *colony* dan *imperialist.*

Dalam membentuk inisialisasi *empire*, *colony* dibagi menjadi beberapa imperial berdasarkan kekuatannya hal ini karena inisialisasi empire harus sebanding dengan kekuatannya. Untuk membagi imperial menjadi beberapa imperial yang sebanding kita mendefinisikan biaya normalisasi *imperialist* yaitu :

$$C_n = c_n - max_i\{c_i\} \quad (2.\ 23)$$

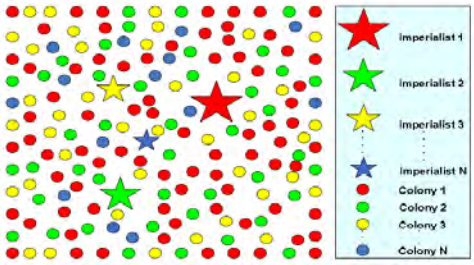
Dimana c_n adalah biaya dari banyaknya n imperialis dan C_n merupakan biaya normalisasi. Normalisasi kekuatan pada tiap-tiap imperialis didefinisikan sebagai :

$$P_n = \left| \frac{c_n}{\sum_{i=1}^{N_{imp}} c_i} \right| \quad (2.\ 24)$$

Dari sudut pandang lain, normalisasi kekuatan dari imperial adalah bagian dari koloni yang seharusnya dipengaruhi oleh imperialis. Kemudian inisialisasi dari jumlah koloni empire sebagai berikut :

$$N.C_n = round\{p_n . N_{col}\} \quad (2.\ 25)$$

Dimana $N.C_n$ adalah inisialisasi jumlah koloni dari n *empire* dan N_{col} adalah jumlah dari keseluruhan koloni. Gambar 2.11. Menunjukan inisisalisasi populasi tiap-tiap *empire*, dari gambar ditunjukan bahwa *empire* yang besar memiliki jumlah koloni yang banyak, *imperialis* 1 memiliki jumlah koloni yang banyak dibandingkan dengan *imperialis* yang lain.



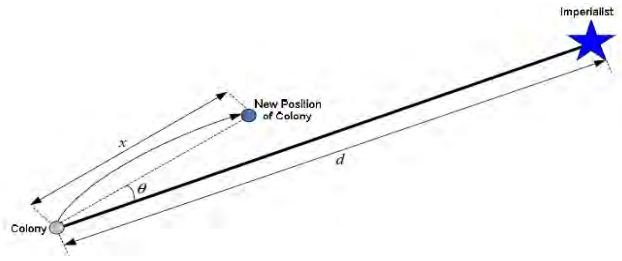
Gambar 2. 11 Membangkitkan inisialisasi *empire* (Atashpaz-Gargari & Lucas, 2007).

b). Pergeseran koloni menuju imperialis

Imperialis country memulai untuk meningkatkan jumlah koloninya yang dimodelkan dengan pergerakan koloni kedalam *imperialis*, pergeseran ini ditunjukan pada Gambar. 2. 12 yang mana koloni bergerak menuju *imperialis* sepanjang arah x. Posisi baru dari *imperialis* ditunjukan dengan warna gelap. Arah pergeseran adalah vektor dari koloni ke *imperialis*. Pada gambar ini, x merupakan variabel acak yang seragam.

$$x \sim U(0, \beta \times d) \qquad (2.\ 26)$$

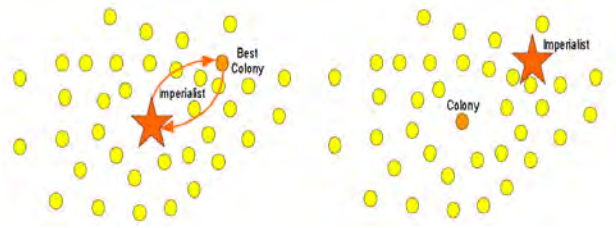
Dimana β bernilai lebih besar daripada 1, dan d adalah jarak antara koloni dengan *imperialis.* Menyebababkan koloni menjadi lebih dekat ke *state imperialis* dari kedua sisi.



Gambar 2. 12 Pergerakan koloni menuju *imperialist*

c). Perubahan posisi imperialis dan koloni

Saat bergerak menuju *imperialis*, koloni dapat menjangkau posisi dengan *lower cost* dibandingkan *imperialis*. Pada kasus ini imperialis bergerak ke posisi koloni dan sebaliknya. Kemudian algoritma akan dilanjutkan *imperialis* pada posisi baru dan kemudian koloni akan bergerak menuju posisi ini. Gambar. 2.13.a menunjukan perubahan posisi antara koloni dan *imperialis.* Pada gambar ini koloni terbaik dari *empire* ditunjukan pada gambar gelap. Koloni ini memiliki *lower cost* daripada *imperialis*. Gambar.2.13.b menunjukan keseluruhan *empire* setelah perubahan posisi dari *imperial* dan koloni.



Gambar 2. 13 (a). Perubahan posisi koloni dan imperialis, (b). Keseluruhan empire setelah perubahan posisi

d). Total power empire32

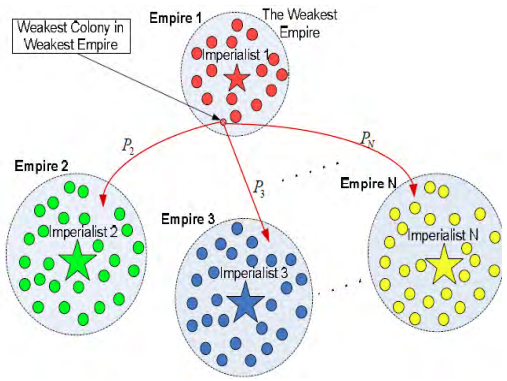
Total power empire berakibat pada kekuatan *imperial country* meskipun kekuatan koloni memberikan efek terhadap *empire*, akan tetapi ini diabaikan pada *total power empire*. Dimodelkan dengan mendefinisikan *total cost* :

$$T.C._n = Cost(imperialis_n) + \xi\ mean\{Cost(colonies\ of\ empire_n)\} \quad (2.27)$$

Dimana total $T.C._n$ adalah *cost* dari n *empire* dan ξ adalah bilangna positif yag ditentukan lebih besar daripada 1. Nilai ξ yang kecil menyebabkan total power dari *empire* lebih dipengaruhi oleh *imperialist* daripada oleh koloni.

e). Kompetisi imperialis

Semua imperialis mencoba untuk memiliki koloni pada empire lain dan mengontrolnya. Kompetisi *emperial* ini berangsur-angsur menurunkan kekuatan empire menjadi lebih lemah dan meningkatkan kekuatan pada yang lainnya. Kekuatan dimodelkan dengan mengambil beberapa (umumnya satu) koloni terlemah dari koloni yang lemah dan membentuk kompetisi diantara semua *empire* untuk memiliki koloni tersebut. Gambar 2.14. Menunjukan model kompetisi imperialis. Berdasarkan total power, pada kompetisi ini tiap-tiap *empire* akan memiliki kemungkinan untuk dapat memiliki koloni yang disebutkan. Dilain sisi koloni tersebut tidak akan dimiliki oleh *empire* terkuat, akan tetapi *empire* akan lebih jauh untuk mendapatkannya.



Gambar 2. 14 Kompetisi Imperialist

Untuk memulai kompetisi, pertama dicari peluang kepemilikan tiap-tiap empire berdasarkan pada total power. Normalisasi total biaya secara sedehana dihitung dengan persamaan

$$N.T.C._n = T.C._n - max_i\{T.C._n\} \quad (2.\ 28)$$

Dimana $T.C._n$ dan $N.T.C._n$ berturut turut adalah *total* cost dan *normalisasi total cost dari n empire*. Dengan menormalisiasi total cost, kemungkinan kepemilikan tiap-tiap empire diberikan pada persamaan

$$P_{p_n} = \left| \frac{N.T.C._n}{\sum_{i=1}^{N_{imp}} N.T.C._n} \right| \quad (2.\ 29)$$

Untuk menentukan koloni yang dipanggil diantara *empire* berdasarkan peluang kepemilikan, dibentuk vektor P pada persamaan

$$P = \left[P_{p_1}, P_{p_2}, P_{p_3}, \dots, P_{pN_{imp}}\right] \quad (2.\ 30)$$

Selanjutnya dibentuk vektor dengan ukuran yang sama sebagai P yang mana jumlah elemennya seragam terdistribusi acak berikut persamaannya

$$R = \left[r_1, r_2, r_3, \dots, r_{N_{imp}}\right] \quad (2.\ 31)$$

$$r_1, r_2, r_3, \dots, r_{N_{imp}} \sim U(0,1)$$

Kemudian vektor D dibentuk dengan memasukkan R kedalam bentuk

$$D = P - R = \left[D_1, D_2, D_3, \dots, D_{N_{imp}}\right] = \left[P_{p_1} - r_1, P_{p_2} - r_2, P_{p_3} - r_3, \dots, P_{pN_{imp}} - r_{N_{imp}}\right] \quad (2.\ 32)$$

Berdasarkna vektor D, (Atashpaz-Gargari & Lucas, 2007) menangani koloni yang dipanggil ke dalam *empire* yang mana indek relevan ada pada D maksimum

f). Eliminasi empire lemah

Empire lemah akan hilang (*collapse*) saat kompetis *imperial* dan koloni akan memisah diantara *empire* lain. Pada mekanisme *collapse* model, perbedaan kriteria dapat digambarkan dengan mempertimbangkan *empire* lemah. Pada sebagian besar implementasinya diasumsikan empire dikolapskan dan dieliminasi ketika dihilangkan semua koloninya.

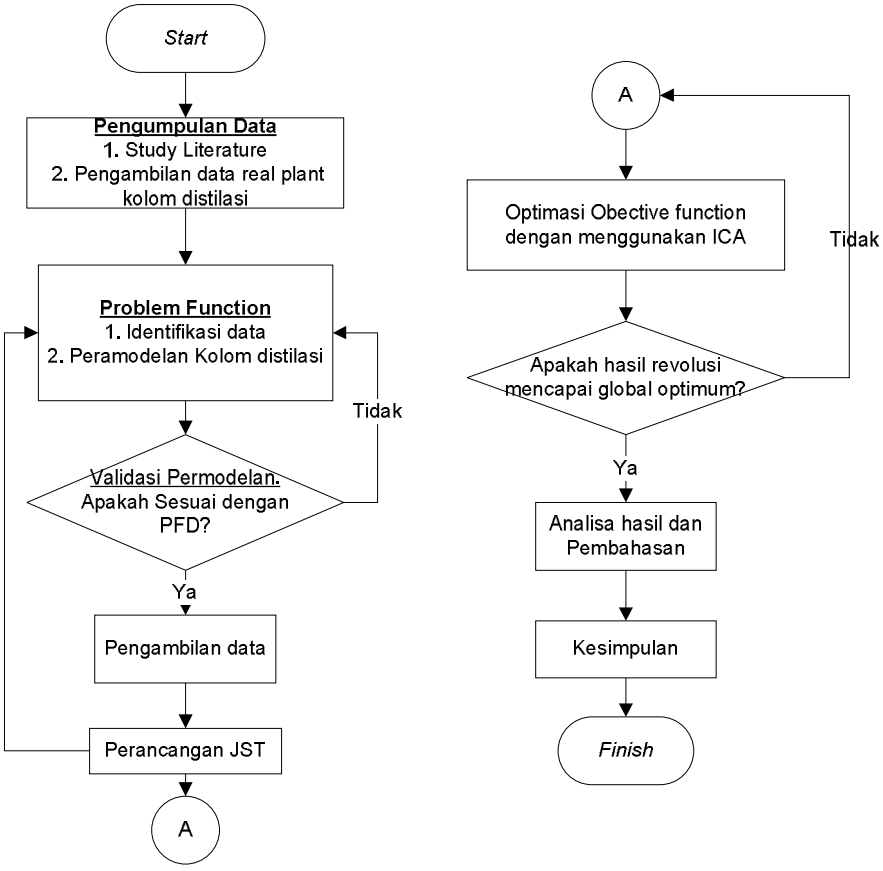
g).Konvergensi

Setelah semua *empire* kecuali yang terkuat akan *collapse* dan semua koloninya akan dikontrol dibawah *empire uniq*. Idealnya semua koloni akan akan memiliki posisi yang sama dan *cost* sendirinya. Idealnya tidak ada perbedaan, tidak hanya diantara koloni tetapi juga diantara koloni dan imperialis. Pada kondisi ini kita dapat mengakiri kompetisi imperial dan menghentikan algoritma

BAB III
METODE PENELITIAN

3.1 *Flowchart* Metodologi Penelitian

Bab ini membahas alur penelitian optimasi kolom distilasi biner dengan menggunakan *Imperial Competitive Algorithm* (ICA). Penelitian ini terdiri dari beberapa bagian dimulai dari studi identifikasi pengambilan data, permodelan kolom distilasi biner, perancangan jaringan syaraf tiruan dan optimasi menggunakan ICA. Metodologi dalam pengerjaan penelitian ini dapat digambarkan dalam bentuk diagram alur sesuai dengan Gambar 3.1 sebagai berikut :



Gambar 3. 1 Diagram Alur Penelitian

3.2 Tahap-tahap penelitian

Penelitian ini terdiri dari beberapa bagian yaitu pengumpulan data, penentuan fungsi permasalahan, validasi permodelan, perancangan jaringan syaraf tiruan (JST)

dan proses optimasi dengan menggunakan *imperialist competitive algorithm* (ICA). Berdasarkan diagram alur pada Gambar 3.1 , Penelitian dilaksanakan dengan rincian masing-masing kegiatan sebagai berikut:

1. Pengumpulan data

   Tahap ini dilakukan pengumpulan informasi terkait mengenai penelitian berupa *study literature*, pengumpulan data proses yang meliputi *Process flow Diagram* (PFD), *Piping & Instrumentation Diagram* (P&ID) dari data *real* plant.

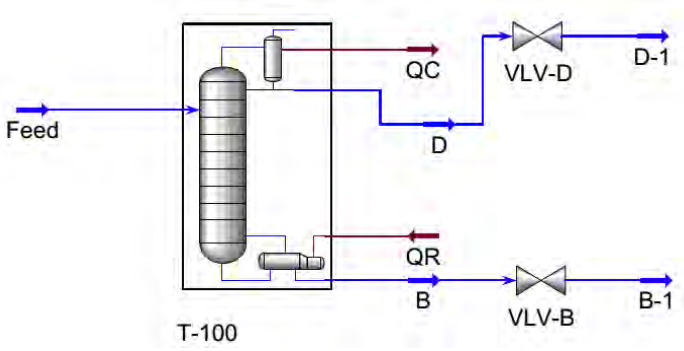
2. *Problem Function*

   a. Identifikasi data

      Data yang diperoleh dari plant selanjutnya diidentifikasi. Selanjutnya dilakukan proses perhitungan seperti yang terlampir pada LAMPIRAN-A. Hasil dari identifikasi bermanfaat untuk mengembangkan rancangan proses sehingga dapat dilakukan proses permodelan dan analisis.

   b. Permodelan kolom distilasi

      Plant yang digunakan dalam penelitian yaitu kolom distilasi biner ethanol-1-propanol dengan menggunakan struktur pengendalian LV (*Liquid-Vapour*).

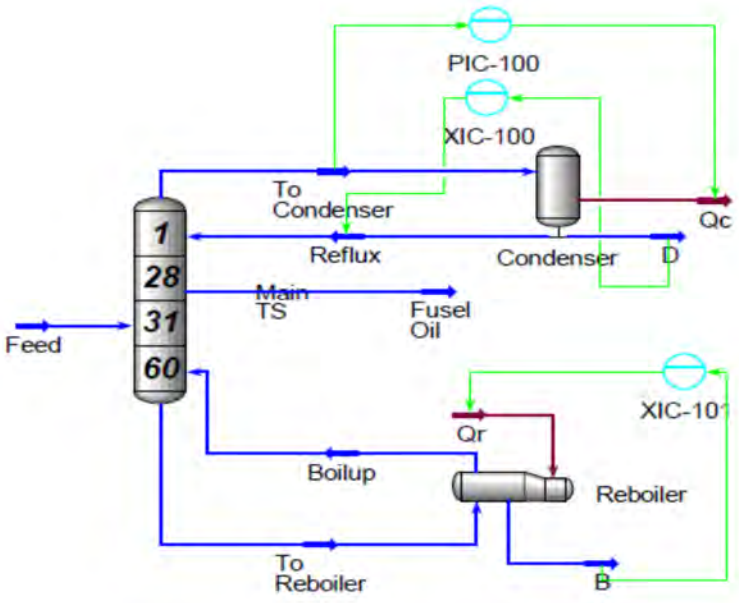


Gambar 3. 2 Model sistem kolom distilasi biner

Gambar 3.2 merupakan gambar model sistem kolom distilasi biner. Permodelan desain proses dilakukan dengan menggunakan aplikasi software aspen HYSYS 7.1 dengan tujuan untuk mendapatkan kondisi *steady state*. Setelah diperoleh keadaan proses yang *steady* maka tahap selanjutnya dilakukan validasi, apabila hasil sesuai dengan data *process*

flow diagram (PFD) maka dapat diambil langkah selanjutnya. Proses validasi ini digunakan sebagai acuan awal yang menunjukan bahwa simulasi dapat mendekati kondisi yang sebenarnya. Apabila validasi dari salah satu tahap tidak dihasilkan output yang memiliki error minimal, maka akan mempengaruhi validasi untuk taha proses berikutnya.

Proses distilasi dilakukan berdasarkan perbedaan titik didih campuran sehingga analisa proses distilasi meliputi hukum kesetimbangan massa, komponen dan energi (*mass and energy balance*). Dari segi komposisi proses distilasi direncanakan pada proses pemisahan ethanol-propanol, serta penentuan optimalisasi kondisi operasi ditentukan pada proses yang membutuhkan energi terbesar yaitu pada laju aliran *reflux*, laju aliran panas pada *reboiler* dan tekanan pada kolom atas. Penetapan parameter meliputi penetapan variabel yang berpengaruh dalam proses optimasi sehingga kemurnian produk dapat diperoleh dan penghematan energy dapat dicapai. Berikut ini Gambar 3.3 permodelan proses beserta kontrollernya.



Gambar 3. 3 Model sistem kolom distilasi biner

Adapun data permodelan mengacu pada data yang diperoleh dari plant bioethanol di PT. Energy Agro Nusantara. Berikut ini Tabel 1. data

spesifikasi perancangan sistem steady-state kolom distilasi ethanol-1-propanol.

Tabel 1. Spesifikasi sistem kolom distilasi

Variabel	**Nilai**
Laju umpan (F), m^3/h	13.50
Temperatur Feed (F), ^oC	101.4
Laju produk Distilat (D), kgmole/h	226.9
Laju produk Bottom (B), kgmole/h	1.598
Komposisi umpan, fraksi mol ethanol (Xf)	0.995790
Komposisi distilat, fraksi mol ethanol (Xd)	0.994335
Komposisi bottom, fraksi mol ethanol (Xb)	0.0001
Jumlah Tray	60
Feed position	31
Tekanan operasi (atm)	1
Beban Condenser (Qc), 10^8 kJ/jam	2.5
Beban Reboiler (Qr), 10^7 kJ/jam	2.018

Setelah memasukan spesifikasi kolom distilasi kedalam permodelan HYSYS dan proses mencapai konvergen, proses pengambilan data dilakukan dengan cara merubah laju input seperti yang terdapat pada LAMPIRAN-B. Data tersebut merupakan cuplikan data yang dihasilkan. Selanjutnya data-data tersebut kemudian dirancang dalam jaringan syaraf tiruan untuk kemudian dioptimasi dengan menggunakan ICA.

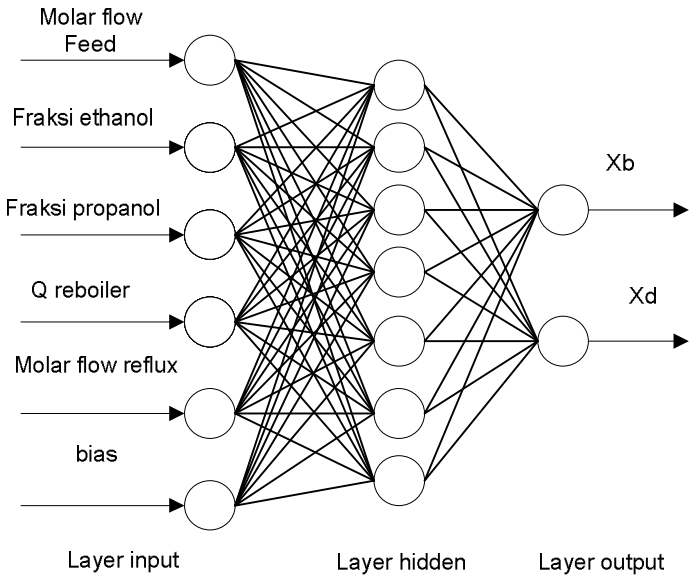
3. Perancangan Jaringan syaraf tiruan

Algoritma identifikasi untuk memodelkan *plant* sistem dijalankan dengan menggunakan jaringan syaraf tiruan, hal ini karena sistem pada JST menyerupai sistem pada kolom distilasi biner dimana terdiri dari input dan output. Proses identifikasi ini perlu dilakukan untuk menentukan variable mana yang dijadikan input dan variable mana yang diadikan output. Variabel input dan output ini umumnya ditentukan dari perannya dalama suatu proses. Variabel input adalah variable yang dimanipulasi dan variabel output adalah variable yang diinginkan (set point). *Disturbance* dari proses sendiri ditentukan dari error proses. Dalam jaringan syaraf tiruan terdapat arsitektur algoritma yang terdiri dari 3 bagian yaitu input layer, hidden layer dan output layer. Dalam jaringan syaraf tiruan digunakan input sebanyak 5, hidden layer sebanyak 20 dan output layer sebanyak 4. Input pada JST berupa komposisi *feed*, laju aliran *feed*, laju aliran *reflux*, dan panas pada *reboiler* serta *condenser*, sedangkan output JST berupa fraksi mol dan fraksi mol produk bawah.

Tabel 2. Input dan output JST

Input	**Output**
Molar flow distilate	Fraksi mol distilat
Mol fraksi ethanol (EtOH)	Fraksi mol bottom
Mol fraksi 1-propanol	
Panas reboiler (Qr)	
Molar flow reflux	

Dalam penerapannya proses optimasi dapat langsung dijalankan dengan menggunakan ICA tanpa perlu dibawa ke JST, akan tetapi dari segi efisiensi waktu (*time consuming*) tidak efektif karna memakan waktu lebih lama.



Gambar 3. 4 Arsitektur jaringan dari plant

4. Optimasi operasi kolom distilasi biner dengan ICA

   Optimasi ICA dilakukan dengan menentukan fungsi objektif, constrain, dan design variable. Optimasi dilakukan dengan menggunakan software MATHLAB R2010a, setelah parameter tersebut ditetapkan. Proses optimasi dengan ICA dapat ditunjukan pada Gambar 3.5. Dan fungsi objektif nya terdapat pada persamaan 3.1 sebagai berikut:

$$J = \max(X_d^* - X_d) + (X_b^* - X_b) \qquad (3.1)$$

   Ket:

   J : Fungsi objektif

   X_d^* : Fraksi mol distilat sesudah (prediksi)

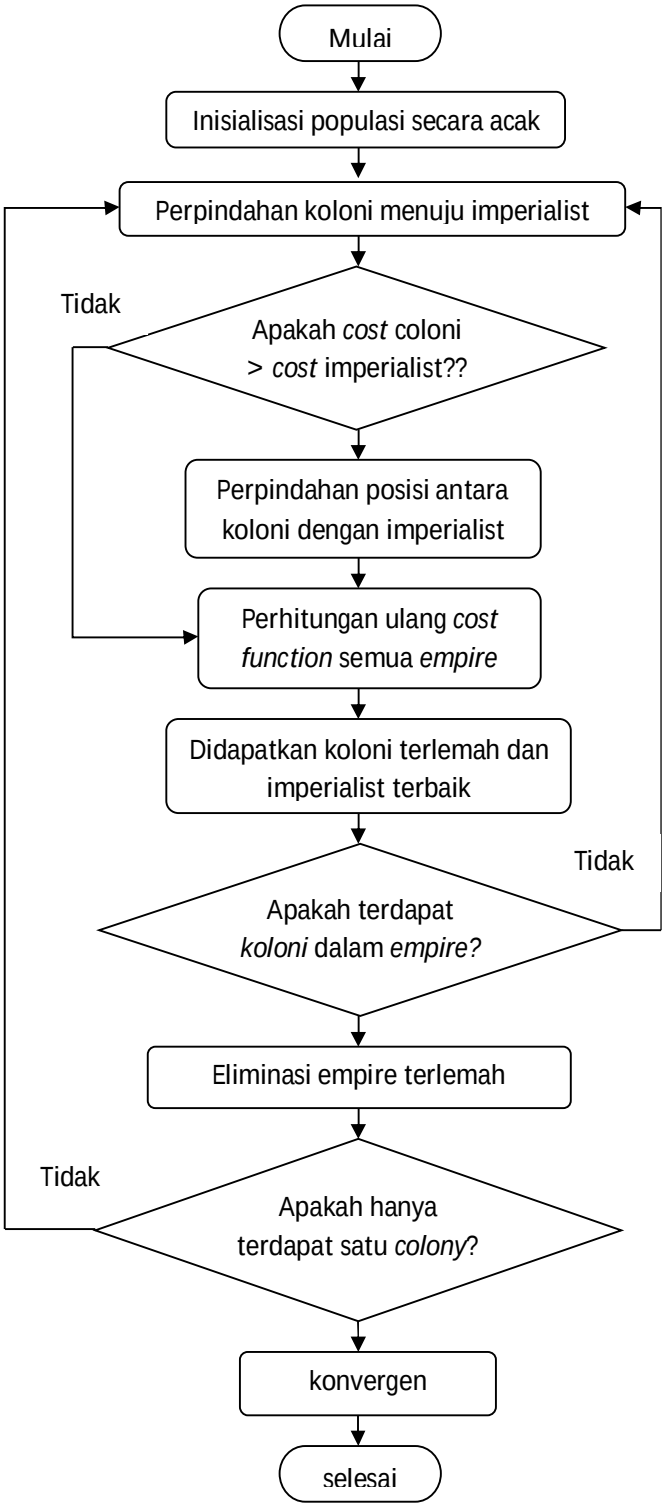
   X_d : Fraksi mol distilat sebelum (target)

   X_b^* : Fraksi mol bottom produk sesudah (prediksi)

   X_b : Fraksi mol bottom produk sebelum (target)

5. Analisa dan kesimpulan

   Merupakan tahap akhir yang dilakukan. Kesimpulan yang dieroleh akan menjawab dari tujuan dan menyelesaikan masalah yang diangkat dalam topik penelitian. Selanjutnya laporan disusun sesuai dengan hasil yang telah dikerjakan.



Gambar 3. 5 Flow Chart ICA

Halaman ini sengaja dikosongkan

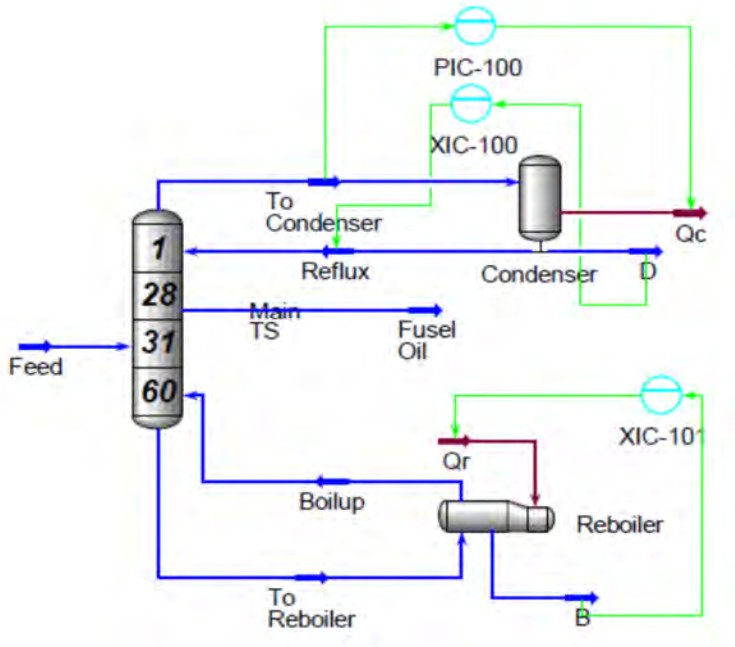
BAB IV
HASIL DAN PEMBAHASAN

Bab ini membahas mengenai hasil optimasi kodisi operasi kolom distilasi biner yang telah disimulasikan dengan menggunakan menggunakan *software* Aspen HYSYS 7.1. Simulasi dilakukan untuk mendapatkan nilai optimum dari komposisi *top product* dan *bottom product* yang sesuai dengan menggunakan *imperialist competitive algorithmn (*ICA). Optimalisasi dilakukan untuk mendapatkan kualitas produk terbaik dimana salah satu parameternya berupa fraksi mol dari komposisi suatu produk. Fraksi mol menunjukan komposisi campuran dengan satuan tak berdimensi yang mana merupakan besarnya jumlah konstituen dibagi besarnya total semua konstituen dan campuran. Hal ini menunjukan ketika suatu konstituen masih mengandung konstituen lain maka komponen tersebut masih dikatakan belum mencapai kemurnian sedangkan kualitas produk ditentukan oleh kadar kemurniannya salah satunya dengan fraksi mol komponen. Besarnya kadar kemurnian produk yang didapatkan berpengaruh terhadap banyaknya energi yang dipakai.

Pada penelitian ini besarnya energi panas reboiler serta laju aliran perlu dikontrol dan dioptimalkan. Pasokan energi yang masuk kedalam reboiler yang terlalu besar menyebabkan proses penguapan yang cepat sehingga fraksi mol *bottom product* ikut menguap ke atas dan *top product* menjadi berlimbah akan tetapi jauh dari kemurnian produk sebab uap *bottom product* bergerak keatas. Banyaknya aliran *reflux* yang masuk ke dalam kolom distilasi yang berlebihan juga dapat mengakibatkan sukarnya proses pemisahan zat. Sehingga pemilihan akan panas reboiler dan laju aliran *reflux* perlu menjadi pertimbangan untuk mengatur kadar kemurnian produk yang dihasilkan.

4.1 Permodelan dinamik kolom distilasi biner

Permodelan dinamik kolom distilasi biner dilakukan dengan mengambil data *real plant* di pabrik bioethanol di PT Energy Agro Nusantara seperti terlampir pada Gambar 4.1.

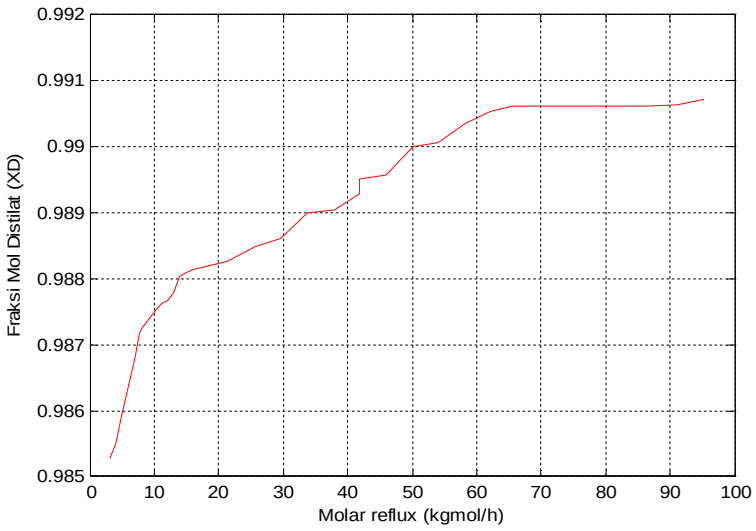


Gambar 4. 1 Permodelan kolom distilasi

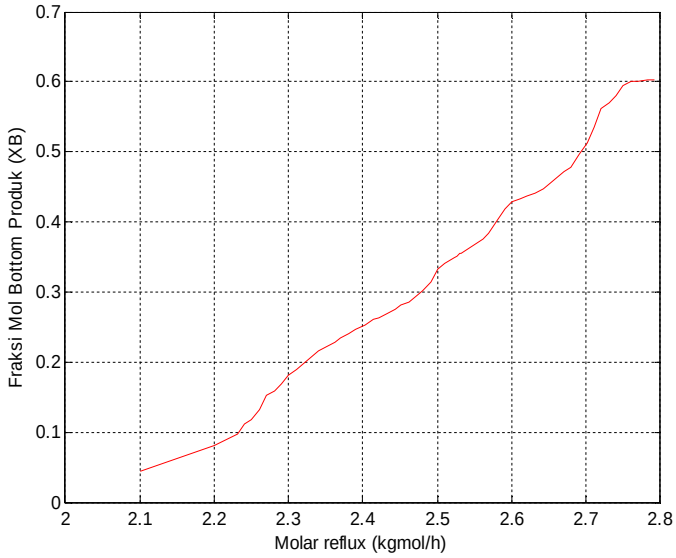
Dalam gambar tersebut terlihat umpan masuk kedalam kolom pada tray nomor 31. Pada Gambar 4.1 diketahui terdapat fusel oil yang digunakan sebagai aliran samping atau *side stream* yang berfungsi untuk mengurangi beban tekanan dalam kolom. Pada permodelan kolom distilasi biner ini digunakan struktur pengendalian L-V (Liquid-Vapour) untuk mengendalikan komposisi produk X_d dan X_b. Pengendalian ini menggunakan aliran *reflux* L untuk mengatur X_d dan Q_r yang masuk ke dalam reboiler digunakan untuk mengatur komposisi vapour yang menguap pada bottom V danX_b. Laju aliran distilat (X_d) dipakai untuk mempertahankan level *reflux* dan laju aliran bottom (X_b) digunakan untuk mengatur level pada kolom distilasi. Proses validasi permodelan kolom distilasi biner dengan HYSYS dilakukan dengan mempertimbangkan prinsip *mass and energy balance.* Dimana hasil input sama dengan hasil output proses.

Dari poses permodelan dinamik diketahui beberapa faktor penting yang berpengaruh terhadap proses, diantaranya adanya perubahan variabel proses dari molar *reflux* dan panas reboiler terhadap fraksi mol distilat dan bottom produk seperti yang terlampir pada Gambar 4.2 sampai Gambar 4.5. Profil fraksi distilat terhadap

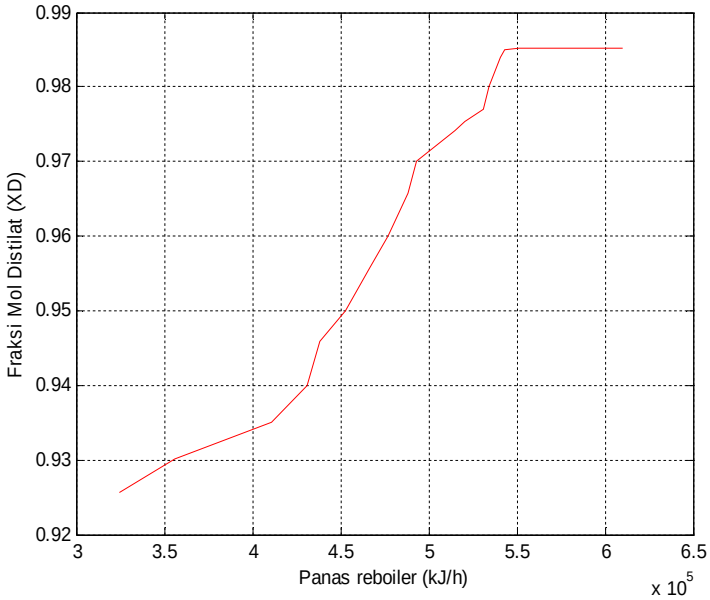
molar *reflux* ditunjukan pada Gambar 4.2. Hasil ini menunjukan bahwa diperoleh adanya peningkatan fraksi distilat disetiap perubahan molar *reflux*. Profil fraksi bottom produk terhadap molar flow *reflux* ditunjukan pada Gambar 4.3. Hasil ini menunjukan bhwa diperoleh adanya peningkatan fraksi bottom produk disetiap penambahan flow *reflux*. Profil fraksi molar distilat terhadap perubahan Q reboiler ditunjukan pada Gambar 4.4. Hasil ini menunjukan bahwa adanya peningkatan molar distilat pada setiap perubahan Q-reboiler. Profil fraksi bottom produk terhadap perubahan Q reboiler ditunjukan pada Gambar 4.5. Hasil ini menunjukan bahwa diperoleh adanya peningkatan bottom produk pada setiap perubahan panas Q reboiler. Hal ini dikarenakan besarnya komposisi pada distilat dan bottom produk dikendalikan oleh laju aliran *reflux* dan panas reboiler, apabila laju aliran *reflux* terlalu besar, maka akan mengakibatkan hilangnya efek pemisahan zat. Hal ini terjadi apabila laju aliran *reflux* terlampau tinggi sedangkan aliran dari *steam reboiler* tetap karena *steam reboiler* merupakan komponen penting dalam perubahan fasa fluida dari cair menjadi gas. Jika pasokan panas lebih sedikit daripada fluida yang harus dipanaskan maka fluida yang berubah fasa menjadi uap juga akan sedikit. Hal ini berpengaruh pada komposisi *top product* yang juga berkurang.



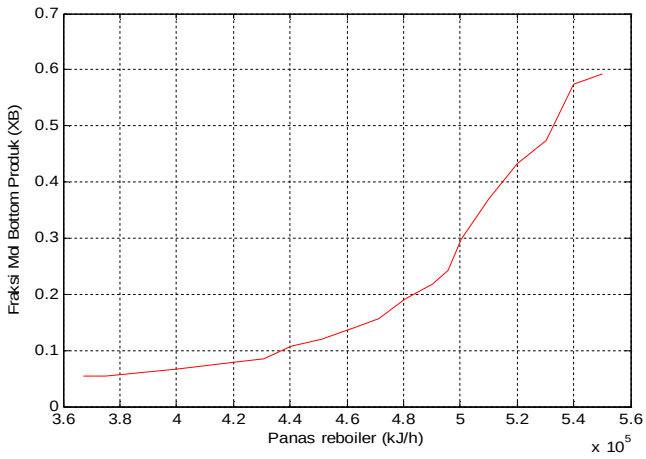
Gambar 4. 2 Profil pengaruh molar *reflux* terhadap komposisi distilat.



Gambar 4. 3 Profil pengaruh molar *reflux* terhadap komposisi bottom produk.

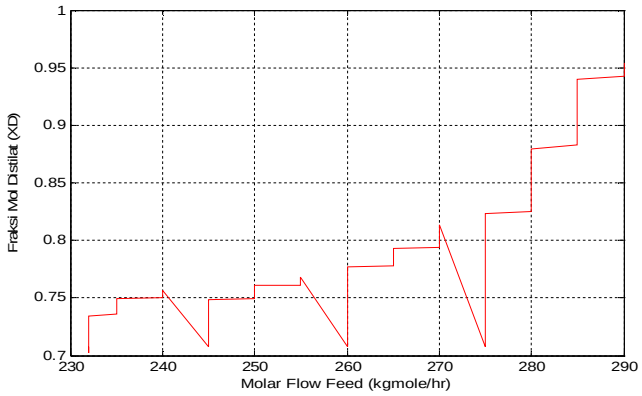


Gambar 4. 4 Profil pengaruh panas reboiler terhadap komposisi distilat.

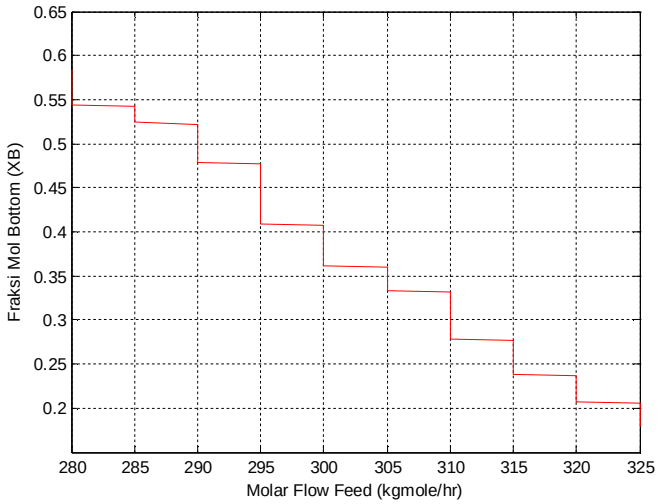


Gambar 4. 5 Profil pengaruh panas reboiler terhadap komposisi bottom produk

Dari keempat grafik diatas menunjukan bahwa besarnya fraksi mol distilat dan bottom produk dipengaruhi oleh laju perubahan *reflux* dan panas reboiler. Perubahan panas reboiler dan *reflux* sebanding dengan perubahan energi, hal ini dikarenakan *reflux* dan panas reboiler merupakan fungsi temperatur. Ketika temperatur meningkat maka energipun semakin meningkat dan titik didih pun meningkat, ketika titik didih meningkat maka komponen akan semakin mudah untuk dipisahkan pada kolom distilasi. Pada Gambar 4.6 merupakan profil perubahan fraksi mol distilat yang meningkat sealan dengan peningkatan laju aliran umpan. Gambar 4.7 Profil pengaruh molar flow *feed* terhadap komposisi bottom produk. Pada Gambar 4.7 besarnya laju aliran bottom produk menurun seiring dengan meningkatnya laju aliran umpan.

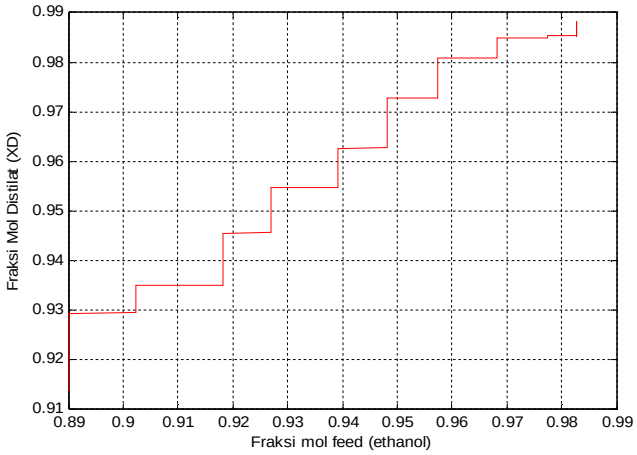


Gambar 4. 6 Profil pengaruh molar flow *feed* terhadap komposisi distilat.

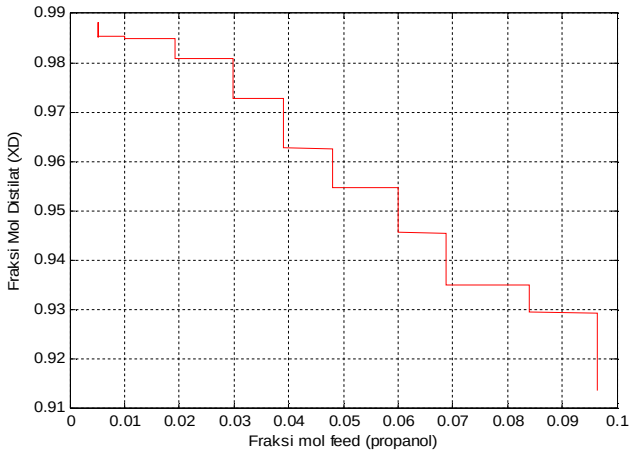


Gambar 4. 7 Profil pengaruh *feed* terhadap komposisi bottom produk.

Pada Gambar 4. 8 terlihat profil perubahan fraksi mol distilat yang semakin meningkat seiring dengan meningkatnya fraksi mol ethanol, sedangkan pada Gambar 4. 9 terlihat profil perubahan fraksi mol distilat yang semakin menurun seiring dengan meningkatnya perubahan fraksi mol ethanol. Kedua gambar ini menjelaskan bahwa besarnya fraksi mol yang dihasilkan pada *top product* (distilat) bergantung pada fraksi mol umpan dan jenis komposisi umpan.



Gambar 4. 8 Profil pengaruh fraksi mol *feed* terhadap fraksi mol distilat.



Gambar 4. 9 Profil pengaruh fraksi mol *feed* terhadap fraksi mol distilat.

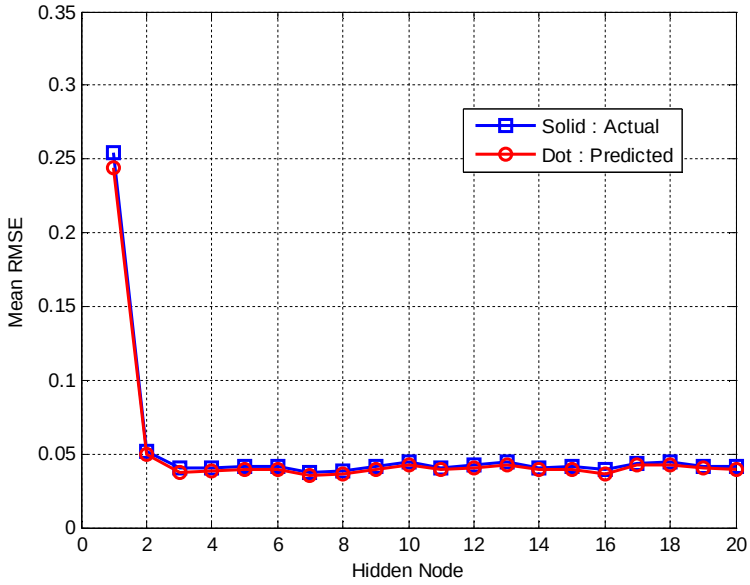
4.2 Perancangan Jaringan Syaraf Tiruan (JST)

Jaringan syaraf tiruan digunakan untuk memodelkan sistem, hal ini karena struktur pada JST menyerupai struktur mimo proses (*multivariable input multivariable output*) proses dari sistem kolom distilasi biner. Peran JST dalam penelitian ini yaitu sebagai jembatan yang digunakan untuk memetakan variabel input dan output proses, dimana inputannya berupa *molar flow feed*, *mole fraction feed* dari ethanol dan 1-propanol, panas kerja reboiler dan *molar flow reflux* dan outputannya berupa fraksi mol distilat dan fraksi mol *bottom product.* Apabila proses variabel ini langsung dijalankan dengan menggunakan ICA maka akan membutuhkan waktu lebih lama disetiap proses iterasinya.

JST menggunakan 80% data training dan 20% data validasi untuk mendapatkan nilai RMSE terbaik dan koefisien koelasi terbaik. Total data yang digunakan adalah 2550 data training dan 650 data validasi. Hal ini dikarenakan JST digunakan sebagai algoritma identifikasi sistem, dimana dalam identifikasi sistem memiliki beberapa kelemahan diantaranya data harus mengikuti kaidah statistika dengan range yang dekat dan jumlah data yang besar sehingga memerlukan waktu pengambilan data yang lebih lama.

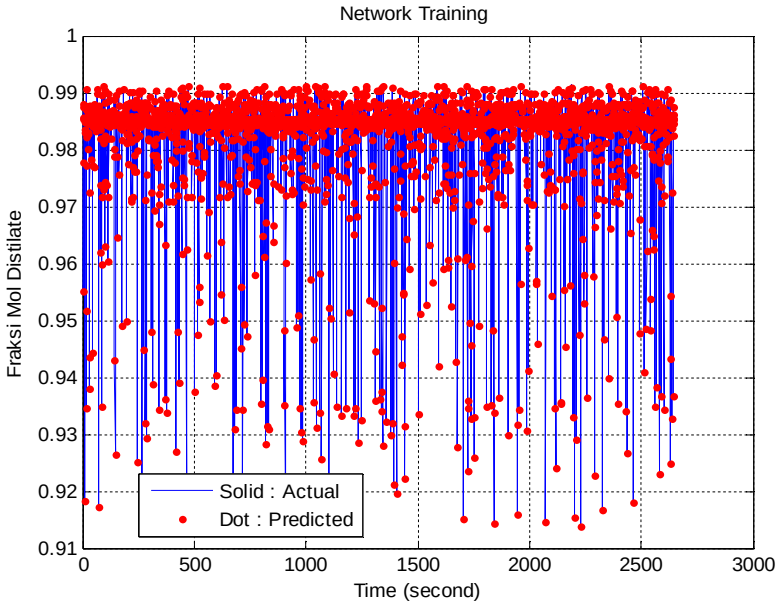
Model kolom distilasi biner dijalankan dengan menggunakan algoritma pembelajaran Lavenberg Maquardt dengan struktur NARX. Hasil JST dengan 3

hidden node sampai 20 hidden node dapat dilihat pada Gambar 4.8. Dari gambar tersebut dapat diambil kesimpulan bahwa JST memiliki *root mean suare error* (RMSE) terkecil pada hidden node ke 7 yaitu sebesar 0,026288. Menurut Cybenko(1989) bahwa semua fungsi kontinu dapat didekati dengan ketelitian yang diinginkan dengan menggunakan jaringan syaraf tiruan satu hidden layer dari hiperbolik tangen hidden neuron dan layer linie pada output neuron. Dari penentuan parameter tersebut maka JST dibuat dengan 20 hidden node dengan 5 masukan dan 2 keluaran JST. Hidden node ke-7 ini selanjutnya digunakan sebagai arsitektur jaringan seperti yang terdapat dalam Gambar 3.4.

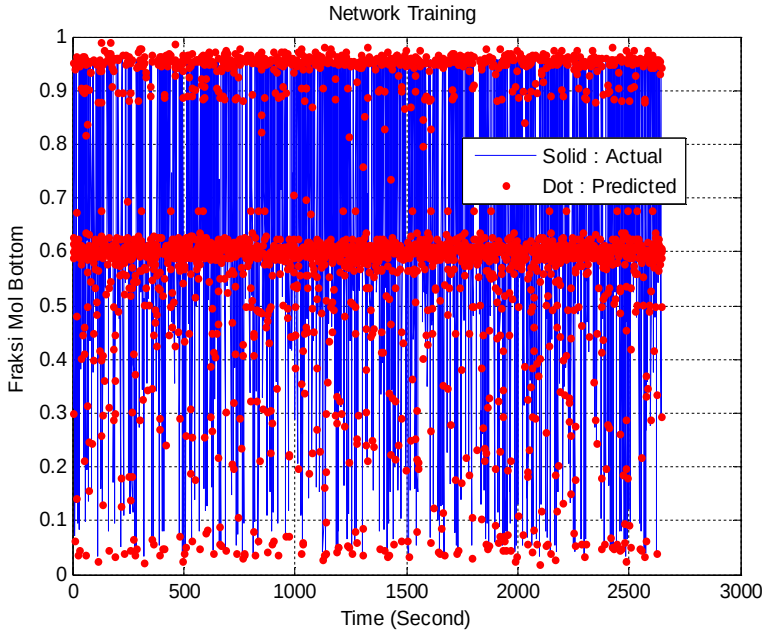


Gambar 4. 10 HN terhadap mean RMSE total

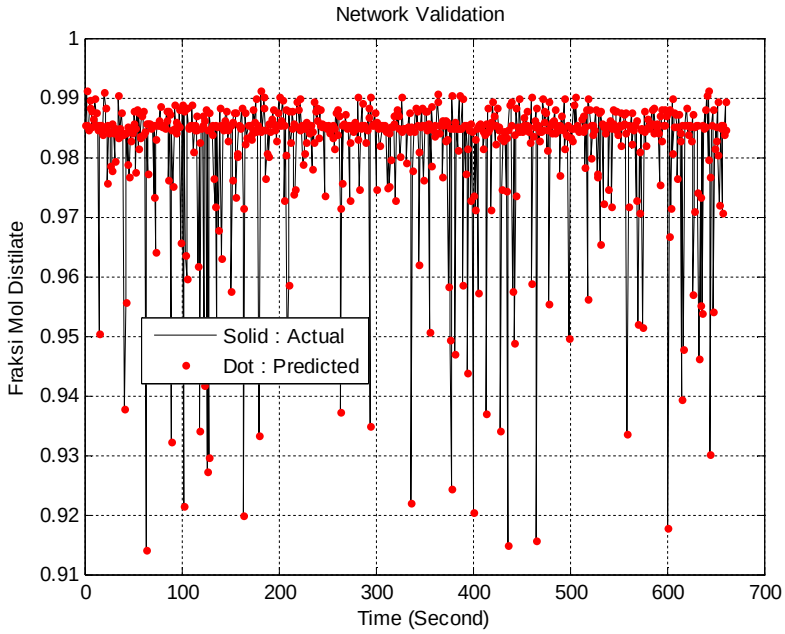
Nilai RMSE ini digunakan untuk mengtahui persentase selisih antara nilai prediksi keluaran dengan nilai set point yang diinginkan. Dari hasil RMSE, JST mampu memberikan hasil yang baik pada permodelan kolom distilasi biner dan mampu menghandle kompleksitas dan non-linieritas dari system. Hasil pelatihan dan validasi kolom distilasi biner dapat dilihat pada Gambar 4.8 sampai Gambar 4.7 sebagai berikut:



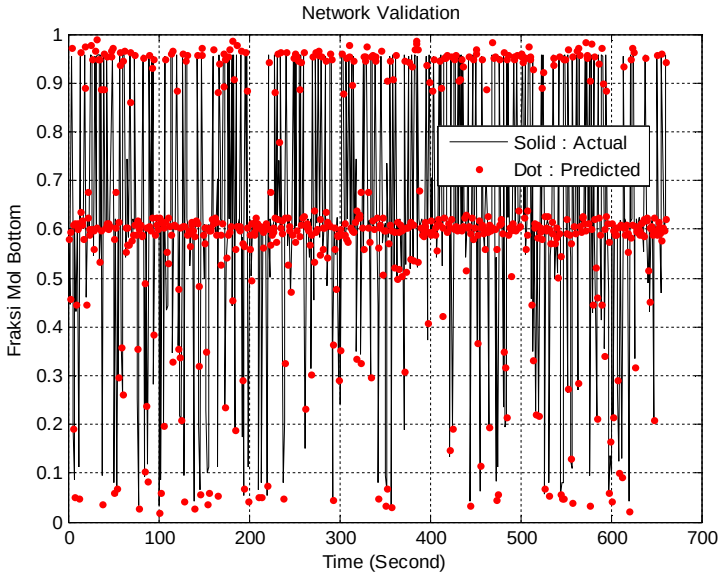
Gambar 4. 11 Hasil pelatihan JST untuk prediksi fraksi mol distilat



Gambar 4. 12 Hasil pelatihan JST untuk prediksi fraksi mol bottom produk



Gambar 4. 13 Hasil validasi JST untuk prediksi fraksi mol distilat



Gambar 4. 14 Hasil validasi JST untuk prediksi fraksi mol bottom produk

4.3 Optimasi dengan menggunakan *Imperialist Competitive Algorithmn* (ICA)

Imperialist Competitive Algoritma digunakan untuk mengoptimasi kolom distilasi biner yang sebelumnya telah dimodelkan dengan menggunakan jaringan

syaraf tiruan (JST). Proses optimasi dijalankan untuk memaksimalkan kualitas produk. Proses asimilasi, revolusi, eliminasi serta kompetisi antar empire pada algoritma ICA menyebabkan tersisa satu imperialist yang terkuat sebagai hasil optimum dari fungsi objective. Dalam proses iterasi beberapa parameter algoritma ICA ditentukan seperti yang terlampir pada Tabel 3, Nilai parameter ICA didasarkan pada penelitian sebelumnya oleh Atazpaz, 2007.

Tabel 3. Parameter algoritma ICA

Parameter ICA	**Nilai**
Jumlah *Country*	200
Jumlah *Imperialist*	8
Laju Revolusi	0.3
Revolusi *(Decade)*	500
Koefisien Asimilasi (β)	2
Assimilation angle coefficient (γ)	0.5
Zeta (ξ)	0.02
Alpha (α)	0.1

Gambar 4.15 menunjukan proses iterasi pada ICA, dimana proses ini dijalankan dengan 500 *decade* atau revolusi. Pada keadaan ini ketika *mean cost* bertemu dengan *minimum cost* di satu titik maka kondisi sudah mencapai solusi optimal artinya semua *colony* sudah menjadi milik *imperialist*, dan *imperialist* tesebut merupakan imperialist terkuat. Validasi ICA dilakukan dengan cara membandingkan hasil performansi paling optimum sesudah dan sebelum proses optimasi. Tabel 4 merupakan perbandingan antara proses sebelum dan sesudah optimasi pada *reflux* dan reboiler. Dari kedua nilai *reflux* dan reboiler yang telah optimal ini kemudian dimasukkan kembali ke dalam JST sebagai nilai prediksi seperti yang ditampilkan pada Tabel 5. Dalam Tabel. 5 hasil JST prediksi yang

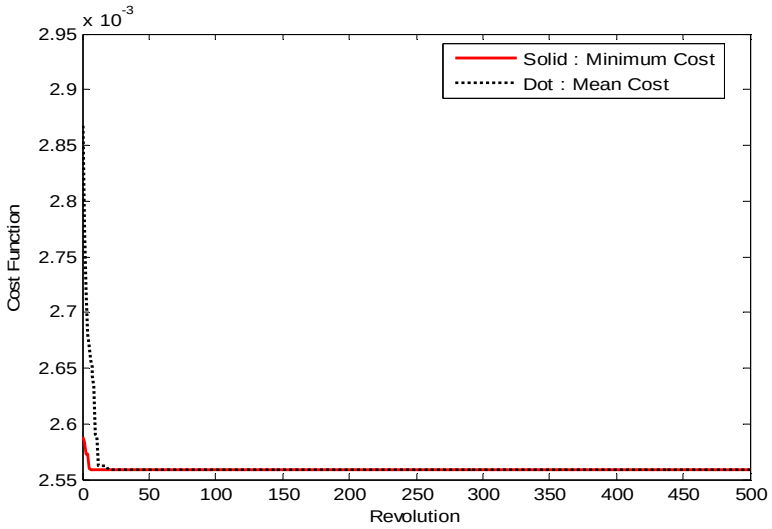
diperoleh dijalankan dalam ICA dengan menggunakan persamaan 3.1 agar diperoleh nilai *cost function* terkecil. *Cost function* mengindikasikan besarnya *error*. Error yang kecil menandakan nilai hasil prediksi mendekati nilai yang sebenarnya atau nilai yang ditargetkan dalam hal ini maka besarnya kemurnian produk yang diperoleh mendekati nilai set point yang diinginkan.

Tabel 4. Perbandingan hasil optimasi

Parameter ICA	**Optimasi dengan ICA**	**Sebelum Optimasi**
Q reboiler	9.0006e+005	892712.5
Reflux	44.4169	41.90798

Tabel 5. Hasil fitness error

Parameter	Target JST	Set point	*Cost Function*
Xd	0.9891	0.99	0.026
Xb	0.007	0.01	



Gambar 4. 15 Proses iterasi dengan ICA

Pada proses optimasi tersebut dilakukan dengan menggunakan 500 *decade*, 200 jumlah *country* dan 8 jumlah *imperialist* sehingga jumlah koloninya sebesar 192, hal ini karena *country* merupakan total penjumlahan banyaknya *colony* dan *imperialist*. Pada data hasil running ICA berikut ini didapatkan beberapa nilai yang dihasilkan ketika jumlah *decade*, *country* dan *imperalist* dirubah. Seperti yang telampir pada Tabel. 6, Tabel 7 dan Tabel 8.

Tabel 6. Hasil *cost function* saat parameter *Country* dirubah

Country	Qr (kJ/h)	R (kgmol/h)	Cost function
250	9.0006e+007	44.4169	0.0026
200	9.0006e+007	44.4169	0.0026
190	9.0006e+007	44.4169	0.0026
180	9.0006e+007	44.4169	0.0026
100	9.0006e+007	44.4169	0.0026
50	9.0006e+007	44.4169	0.0026

Tabel 7. Hasil *cost function* saat parameter *Imperialist* dirubah

Imperialist	Qr (kJ/h)	R (kgmol/h)	Cost function
8	9.0006e+007	44.4169	0.0026
6	9.0006e+007	44.4169	0.0026
4	9.0006e+007	44.4169	0.0026
2	9.0006e+007	44.4169	0.0026

Tabel 8. Hasil *cost function* saat parameter *Decade* dirubah

Decade	Qr (kJ/h)	R (kgmol/h)	Cost function
400	9.0006e+007	44.4169	0.0026
300	9.0006e+007	36.7221	0.0026
200	9.0006e+007	30.2547	0.0026

Pada Tabel. 6 dan Tabel. 7 adanya perubahan pada *imperialist* dan *country* tidak mempengaruhi perubahan nilai panas reboiler dan laju aliran *reflux* terbaik serta nilai *cost function*, akan tetapi pada Tabel. 8 ketika jumlah decade diubah terdapat perubahan pada nilai *reflux*. Hal ini dikarenakan nilai laju aliran *reflux* yang diperoleh sensitif terhadap perubahan proses iterasi yang dijalankan. Grafik dari proses tersebut dicantumkan pada LAMPIRAN C. Dari grafik tersebut terlihat bahwa semakin banyak proses iterasi maka hasil menjadi konvek atau hanya memiliki satu minimum lokal dan ini sesuai dengan pernyataan Atazpaz (2007) bahwa penggunaan jumlah koloni yang banyak dapat menyebabkan tidak effisien karena dapat terjadi global optimum, proses iterasi yang panjang dan hasil menjadi konvek, menurutnya jumlah iterasi yang sesuai sebesar 10 % dari jumlah *country*.

LAMPIRAN A

- **Data Feed**

Feed = Material S 205 + impuritas = Ethanol + impuritas

Impuritas =

1. Ethanol (EtOH)
2. Methanol (MeOH)
3. 3 Metil 1 Butanol (3M1B)
4. 1-Propanol
5. Ethyl Acetate
6. 2 Metil 1 propanol (2M1P)
7. 2 Metil 1 Butan (2M1B)

a. Ethanol (CH3CH2OH)

Massa Molar CH3CH2OH = 12.0107 + (1.00794*3) + 12. 0107 + 1.00794*2 + 15.9994 + 1.00794

= 46.06844 mol/gr

Ethanol Density = 0.7893 gr/cm^3 = 789.3 gr/liter

$$\text{Massa} = \frac{789.3 \times 17.234}{100} = 135.97626$$

$$\text{Mol} = \frac{\text{Massa}}{\text{Massa Molar}} = \frac{135.97626}{46.06844} = 2.951614$$

b. Impuritas EtOH = ND = Non-Defined

c. Impritas MeOH = ND = Non-Defined

d. Impuritas 3 Metil 1 Butanol (3M1B)

Konsentrasi 3M1B = 0.0623

3M1B Density = 0.6213

Molecular weight = 70.1329

$$Mol = \frac{0.0623 \times \frac{1000}{10} \times 0.6213}{70.1329} = 0.0055342$$

e. Impuritas 1-Propanol

Konsentrasi 1-Propanol = 0.0378

1-Propanol Density = 0.8053

Molecular weight = 60.09502

$$Mol = \frac{0.0378 \times \frac{1000}{10} \times 0.8053}{60.09502} = 0.005065$$

f. Impuritas Ethyl Acetate = ND = Non-Defined

g. Impuritas 2 Metil 1 Propanol (2M1P)

Konsentrasi 2M1P = 0.0156

2M1P Density = 0.802

Molecular weight = 74.122

$$Mol = \frac{0.0156 \times \frac{1000}{10} \times 0.802}{74.122} = 0.0016879$$

h. Impuritas 2 Metil 1 Butanol (2M1B) = ND = Non-Defined

- **<u>Data Perhitungan Hysys</u>**

a. Pressure Feed

Letak Feed = Tray 31

Jarak Tray 1-Tray 31 = 30

Pressre drop x jarak = 0.4 x 30 = 12 kPaG

P Tray 1 = 105 kPaG

T Tray 1 = 97.6 C

T Tray 60 = 122.5 C

Dari rumus gas ideal = $PV = nRT$, semakin tinggi nilai T maka semakin tinggi pula nilai P

Sehingga nilai P Tray 31 = 105 kPaG + 12 kPaG

= 117 kPaG

b. Temperatur Feed

Digunakan rumus perbandingan

T Tray 1 ➔ 97.6 C

T Tray 28 ➔ 101 C

T Tray 31 ➔ ???

$$\text{Kenaikan T tiap tray} = \frac{101 - 97.6}{27} = 0.1259259\ ^{\circ}C$$

$$\text{Total kenaikan T dr Tray 1} - \text{Tray 31} = 0.1259259 \times 30 = 3.7777\ ^{\circ}C$$

$$\text{T tray awal} + \text{T total kenaikan tray} = 3.77 + 97.6 = 101.377778\ ^{\circ}C$$

c. Pressure reboiler = P proses = P alat

Selisih tray = 60-1 = 59

Pressure Drop = 0.4

P tray 1 = 105

$$\text{Pressure drop x jarak} = 0.4 \times 59 = 23.6$$

P Tray 60 = P Reboiler = 23. 6 + 105 = 128.6

Halaman ini sengaja dikosongkan

LAMPIRAN B

INPUT					OUTPUT	
MolarFeed	XEtOH	Xprop	*Reflux*	Qr	XD	XB
232.1668	0.982666	5.08E-03	2.742971	535630.2	0.985285	0.62313
232.1668	0.982666	5.08E-03	2.743049	535464	0.985285	0.623099
232.1668	0.982666	5.08E-03	2.743169	534816.7	0.985284	0.623067
232.1668	0.982666	5.08E-03	2.743023	534388.1	0.985285	0.623035
232.1668	0.982666	5.08E-03	2.743565	534186.9	0.985282	0.623004
232.1668	0.982666	5.08E-03	2.743654	534040.8	0.985282	0.622972
232.1668	0.982666	5.08E-03	2.743576	533978.6	0.985282	0.62294
232.1668	0.982666	5.08E-03	2.743286	533940.6	0.985284	0.622908
232.1668	0.982666	5.08E-03	2.743553	533954.3	0.985282	0.622876
232.1668	0.982666	5.08E-03	2.744134	533977	0.985279	0.622844
232.1668	0.982666	5.08E-03	2.744425	533933.1	0.985278	0.622812
232.1668	0.982666	5.08E-03	2.744722	533930.1	0.985276	0.622779
232.1668	0.982666	5.08E-03	2.74514	533919.1	0.985274	0.622747
232.1668	0.982666	5.08E-03	2.74542	533893.4	0.985273	0.622714
232.1668	0.982666	5.08E-03	2.745386	533887.9	0.985273	0.622681
232.1668	0.982666	5.08E-03	2.745279	533915.5	0.985274	0.622648
232.1668	0.982666	5.08E-03	2.745276	533953.2	0.985274	0.622615
232.1668	0.982666	5.08E-03	2.74579	534002.1	0.985271	0.622582
232.1668	0.982666	5.08E-03	2.746045	534043	0.98527	0.622548
232.1668	0.982666	5.08E-03	2.74581	534100.3	0.985271	0.622514
232.1668	0.982666	5.08E-03	2.745521	534192.4	0.985272	0.62248
232.1668	0.982666	5.08E-03	2.746009	534319.7	0.98527	0.622446
232.1668	0.982666	5.08E-03	2.745691	534459.9	0.985272	0.622411
232.1668	0.982666	5.08E-03	2.745361	534632.6	0.985273	0.622375
232.1668	0.982666	5.08E-03	2.745478	534856.6	0.985273	0.622339
232.1668	0.982666	5.08E-03	2.745641	535082.8	0.985272	0.622302
232.1668	0.982666	5.08E-03	2.745508	535313.1	0.985272	0.622265
235	0.982666	5.08E-03	2.745479	503281.5	0.985273	0.622228

235	0.982666	5.08E-03	2.74609	453453.1	0.98527	0.622189
235	0.982666	5.08E-03	2.746783	420920	0.985266	0.622135
235	0.982666	5.08E-03	2.74753	395961.4	0.985262	0.622068
235	0.982666	5.08E-03	2.748196	375188.3	0.985259	0.621985
235	0.982666	5.08E-03	2.748896	356952.8	0.985256	0.621883
235	0.982666	5.08E-03	2.749398	340848.3	0.985253	0.621765
235	0.982666	5.08E-03	2.749543	326361.8	0.985252	0.621627
235	0.982666	5.08E-03	2.749943	313592	0.98525	0.621468
235	0.982666	5.08E-03	2.750065	302301.9	0.98525	0.621286
235	0.982666	5.08E-03	2.75035	292523.8	0.985248	0.62108
235	0.982666	5.08E-03	2.75038	284262.8	0.985248	0.620849
235	0.982666	5.08E-03	2.750388	277784.7	0.985248	0.620587
235	0.982666	5.08E-03	2.750927	274181.6	0.985245	0.620281
235	0.982666	5.08E-03	2.75116	276013.2	0.985244	0.619893
235	0.982666	5.08E-03	2.75134	289328.3	0.985243	0.619304
235	0.982666	5.08E-03	2.751077	324443.5	0.985245	0.618328
235	0.982666	5.08E-03	2.751345	392915.5	0.985243	0.616657
235	0.982666	5.08E-03	2.751151	463378.7	0.985244	0.615478
235	0.982666	5.08E-03	2.750884	514398	0.985246	0.614739
235	0.982666	5.08E-03	2.750726	545782.5	0.985246	0.614199
235	0.982666	5.08E-03	2.750833	565816.9	0.985246	0.613825
235	0.982666	5.08E-03	2.751032	579157.1	0.985245	0.613572
235	0.982666	5.08E-03	2.750602	588660.4	0.985247	0.613403
235	0.982666	5.08E-03	2.750357	595775	0.985248	0.613287
235	0.982666	5.08E-03	2.750071	601285.6	0.98525	0.613205
240	0.982666	5.08E-03	2.74977	600286.3	0.985251	0.613143
240	0.982666	5.08E-03	2.749601	579285.1	0.985252	0.612961
240	0.982666	5.08E-03	2.749481	569155.9	0.985253	0.612672
240	0.982666	5.08E-03	2.749551	565048.5	0.985252	0.612333
240	0.982666	5.08E-03	2.749655	563701.2	0.985252	0.611964
240	0.982666	5.08E-03	2.749606	563561.4	0.985252	0.611574
240	0.982666	5.08E-03	2.749453	563125	0.985253	0.611169

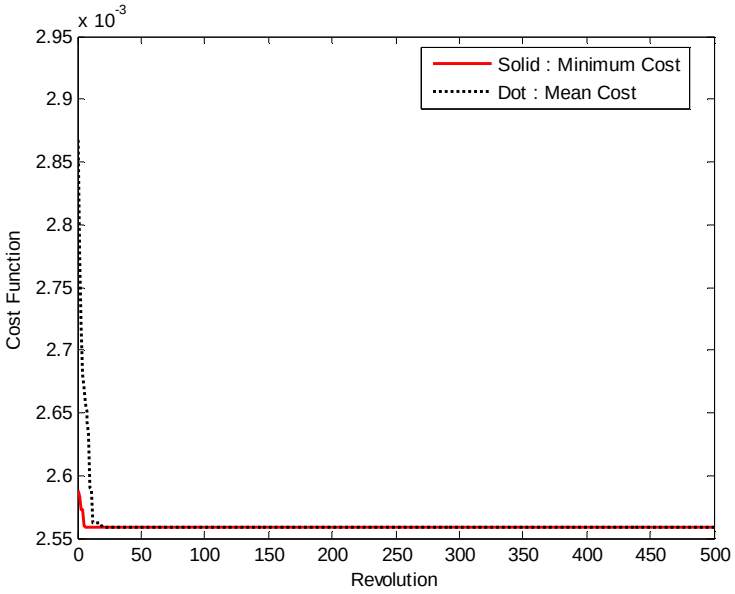
240	0.982666	5.08E-03	2.749424	562028.4	0.985253	0.610763
240	0.982666	5.08E-03	2.749518	560974.2	0.985252	0.610347
240	0.982666	5.08E-03	2.75007	560336.4	0.98525	0.609907
240	0.982666	5.08E-03	2.750199	559942.5	0.985249	0.609462
240	0.982666	5.08E-03	2.750258	559859.4	0.985249	0.609003
240	0.982666	5.08E-03	2.750604	559961.8	0.985247	0.608529
240	0.982666	5.08E-03	2.750436	560199.9	0.985248	0.608042
240	0.982666	5.08E-03	2.750968	560507.5	0.985245	0.607544
240	0.982666	5.08E-03	2.751742	560824.5	0.985241	0.607036
240	0.982666	5.08E-03	2.752217	560990.8	0.985239	0.606521
240	0.982666	5.08E-03	2.752453	561038.2	0.985238	0.606
240	0.982666	5.08E-03	2.752438	560974.8	0.985238	0.605474
240	0.982666	5.08E-03	2.752846	560881.4	0.985236	0.604931
240	0.982666	5.08E-03	2.753099	560660.8	0.985235	0.604397
240	0.982666	5.08E-03	2.753364	560412.6	0.985233	0.603861
240	0.982666	5.08E-03	2.75341	560119.9	0.985233	0.603321
240	0.982666	5.08E-03	2.753846	559796.5	0.985231	0.602779
240	0.982666	5.08E-03	2.754496	559462.3	0.985228	0.602234
245	0.982666	5.08E-03	2.754733	628999.5	0.985226	0.601781
245	0.982666	5.08E-03	2.754691	714165.8	0.985227	0.601705
245	0.982666	5.08E-03	2.755191	743184.7	0.985224	0.601678
245	0.982666	5.08E-03	2.754792	754099.9	0.985226	0.601654
245	0.982666	5.08E-03	2.755001	765513.7	0.985225	0.601628
245	0.982666	5.08E-03	2.754995	777541.9	0.985225	0.601602
245	0.982666	5.08E-03	2.75528	789633.8	0.985224	0.601574
245	0.982666	5.08E-03	2.755693	801192.1	0.985222	0.601545
245	0.982666	5.08E-03	2.75593	812101.7	0.98522	0.601516
245	0.982666	5.08E-03	2.756405	822230.5	0.985218	0.601484
245	0.982666	5.08E-03	2.756253	831696.8	0.985219	0.601452
245	0.982666	5.08E-03	2.756689	840250.2	0.985217	0.601419
245	0.982666	5.08E-03	2.757092	848060.1	0.985215	0.601384
245	0.982666	5.08E-03	2.757548	855086.2	0.985212	0.601348

245	0.982666	5.08E-03	2.758215	861373.1	0.985209	0.60131
245	0.982666	5.08E-03	2.758288	866876.6	0.985209	0.601272
245	0.982666	5.08E-03	2.758239	871656.1	0.985209	0.601232
245	0.982666	5.08E-03	2.758291	875807.6	0.985209	0.601192
245	0.982666	5.08E-03	2.757908	879368.7	0.98521	0.60115
245	0.982666	5.08E-03	2.757665	882317	0.985212	0.601106
245	0.982666	5.08E-03	2.757289	884733.5	0.985214	0.601062
245	0.982666	5.08E-03	2.757194	886578.1	0.985214	0.601017
245	0.982666	5.08E-03	2.756901	887936.7	0.985215	0.60097
245	0.982666	5.08E-03	2.756712	888812.3	0.985216	0.600922
245	0.982666	5.08E-03	2.756181	889280	0.985219	0.600872
245	0.982666	5.08E-03	2.755294	889353.9	0.985224	0.600822
250	0.982666	5.08E-03	2.755183	848219	0.985224	0.600772
250	0.982666	5.08E-03	2.755351	684958.5	0.985223	0.600753
250	0.982666	5.08E-03	2.756332	589385.2	0.985218	0.600766
250	0.982666	5.08E-03	2.757288	533151.1	0.985214	0.600807
250	0.982666	5.08E-03	2.758216	491125.4	0.985209	0.600897
250	0.982666	5.08E-03	2.758996	454220.5	0.985205	0.601059
250	0.982666	5.08E-03	2.759572	423409.8	0.985202	0.601305
250	0.982666	5.08E-03	2.76005	401593.2	0.9852	0.601629
250	0.982666	5.08E-03	2.761296	392069.5	0.985194	0.601986
250	0.982666	5.08E-03	2.761778	396683.4	0.985191	0.602257
250	0.982666	5.08E-03	2.761532	426770.2	0.985192	0.602315
250	0.982666	5.08E-03	2.760755	459534.6	0.985196	0.602294
250	0.982666	5.08E-03	2.759973	480785.1	0.9852	0.602263
250	0.982666	5.08E-03	2.759206	495151.8	0.985204	0.60222
250	0.982666	5.08E-03	2.758422	505722.5	0.985208	0.602168
250	0.982666	5.08E-03	2.757684	514069.1	0.985212	0.602108
250	0.982666	5.08E-03	2.757163	520933.8	0.985214	0.602044
250	0.982666	5.08E-03	2.756609	526704.5	0.985217	0.601979
250	0.982666	5.08E-03	2.755787	531591.9	0.985221	0.601917
250	0.982666	5.08E-03	2.755044	535754.3	0.985225	0.601857

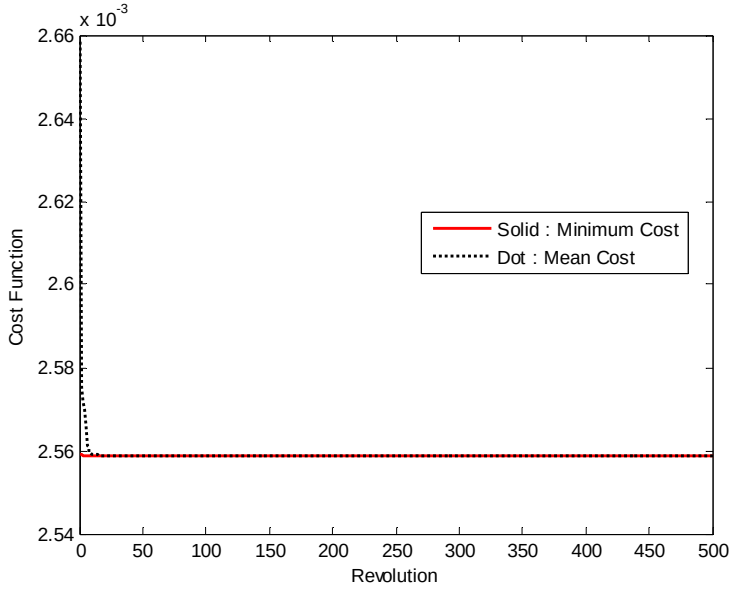
250	0.982666	5.08E-03	2.754085	539271.4	0.98523	0.601801
250	0.982666	5.08E-03	2.753582	542252.1	0.985232	0.601749
250	0.982666	5.08E-03	2.752196	544735.8	0.985239	0.601702
250	0.982666	5.08E-03	2.751542	546846.2	0.985242	0.601659
250	0.982666	5.08E-03	2.750557	548571.4	0.985247	0.601621
255	0.982666	5.08E-03	2.749718	538076.2	0.985251	0.601565
255	0.982666	5.08E-03	2.748383	527902.3	0.985258	0.601378
255	0.982666	5.08E-03	2.747282	526316.3	0.985264	0.601076
255	0.982666	5.08E-03	2.746038	528895.3	0.98527	0.600683
255	0.982666	5.08E-03	2.745336	533278.2	0.985273	0.600217
255	0.982666	5.08E-03	2.743751	538226.7	0.985281	0.599692
255	0.982666	5.08E-03	2.742644	543175.2	0.985287	0.599118
255	0.982666	5.08E-03	2.741561	547798.8	0.985292	0.598505
255	0.982666	5.08E-03	2.740147	551993.3	0.985299	0.597861
255	0.982666	5.08E-03	2.738387	555737.6	0.985308	0.597191
255	0.982666	5.08E-03	2.737116	557710.5	0.985314	0.596509
255	0.982666	5.08E-03	2.735804	557932.2	0.985321	0.59583
255	0.982666	5.08E-03	2.734561	557610.4	0.985327	0.595147
255	0.982666	5.08E-03	2.733138	557143.1	0.985334	0.594454
255	0.982666	5.08E-03	2.731931	556692.9	0.98534	0.593748
255	0.982666	5.08E-03	2.730897	556364.5	0.985346	0.593028
255	0.982666	5.08E-03	2.72974	629822.5	0.985351	0.592438

LAMPIRAN C

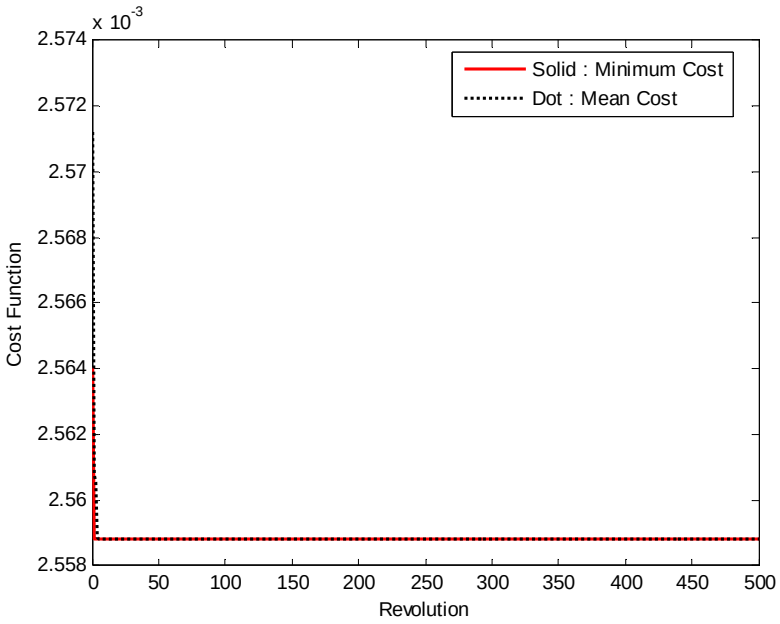
Country 100



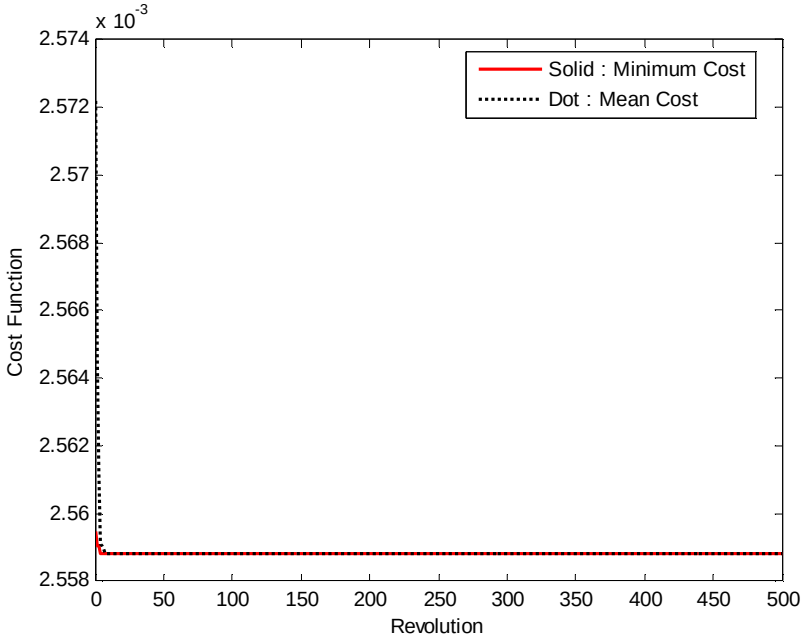
Country 50



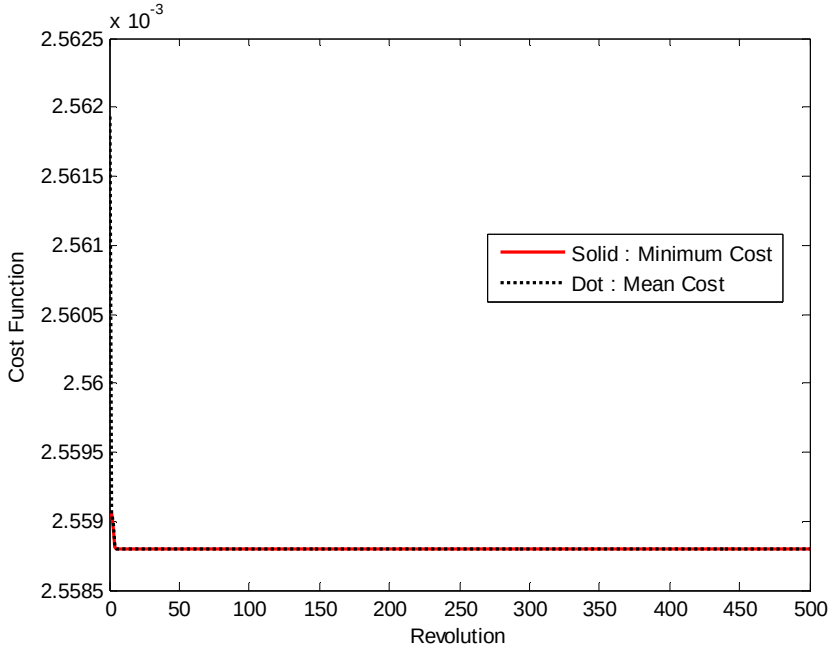
Imperialist 6



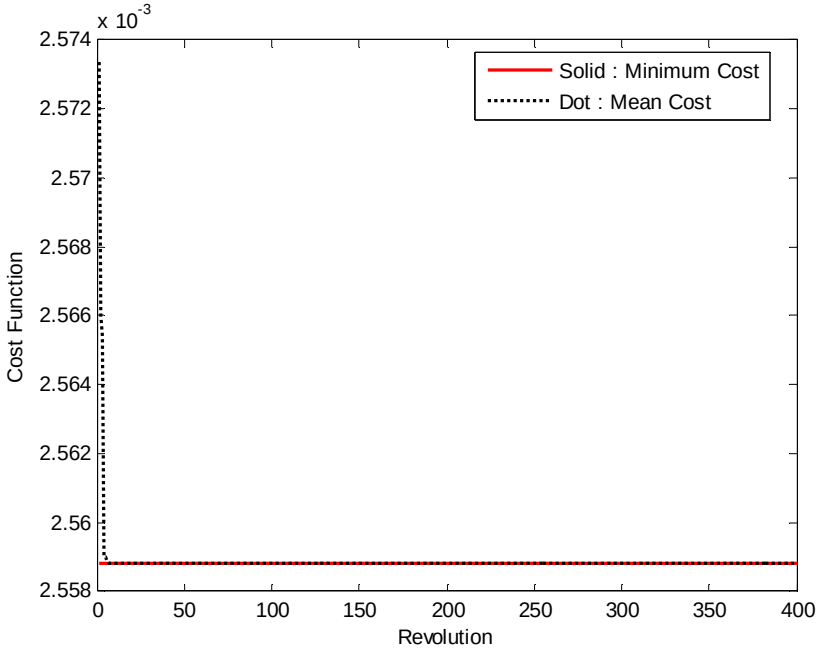
Imperialist 4



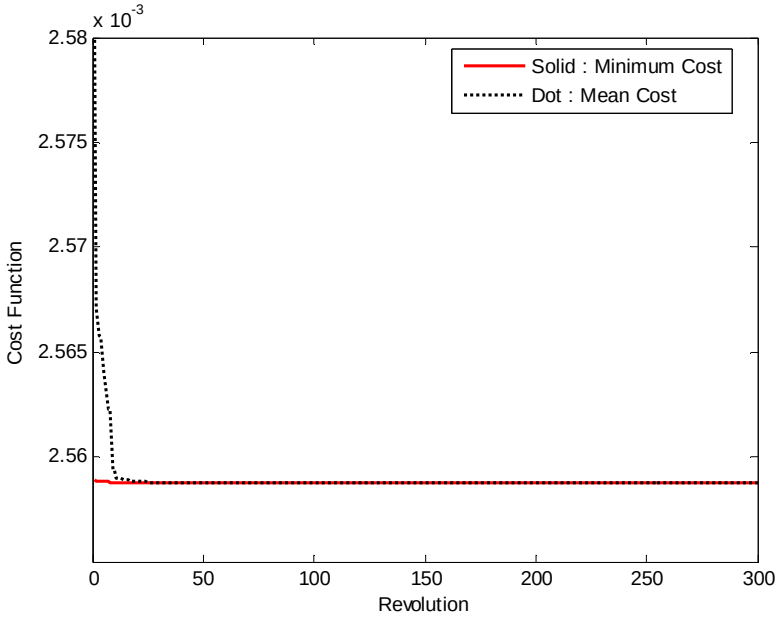
Imperialis 2



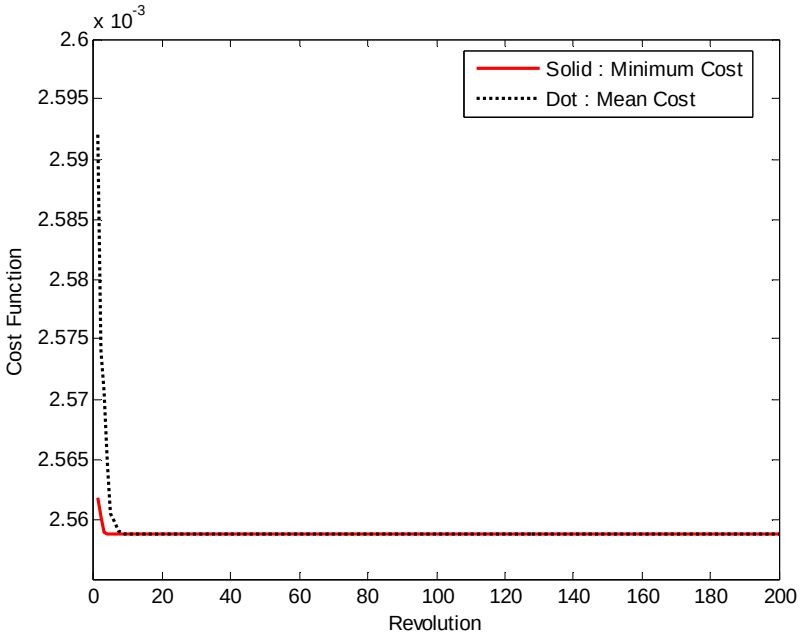
Decade 400



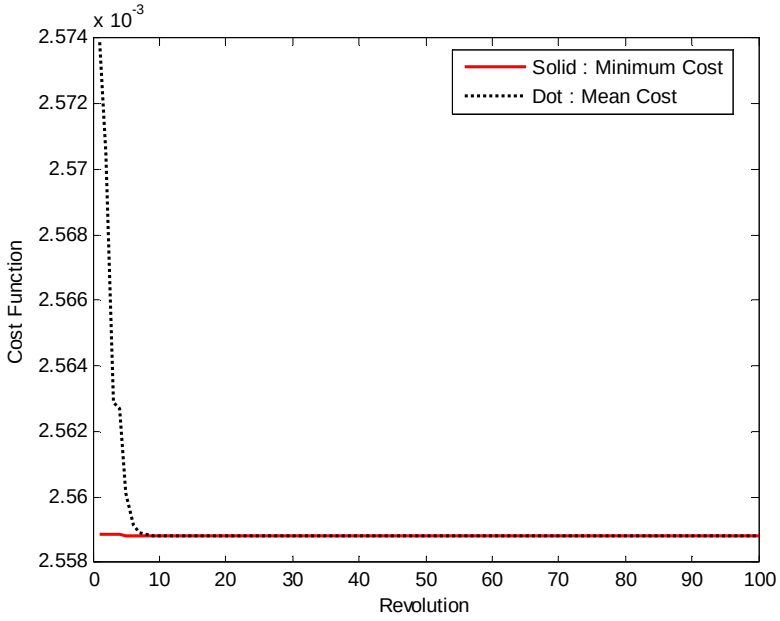
Decade 300



Decade 200



Decade 100



LAMPIRAN D

Proses Training JST

```
clc
close all;
clear all;
clc
disp('-------------------------')
disp('  TRAINING IN PROGRESS   ')
disp('-------------------------')
A = xlsread('DATAASELI.xlsx','RandomJST');
[rowTr,colTr] = size(A);

ut = A(2:1927,1:5)'; % Data training input
yt = A(2:1927,6:7)'; % Data training output

uv  = A(1928:2408,1:5)'; % Data validasi input
yv = A(1928:2408,6:7)'; % Data validasi output

us = A(2:2408,1:5)'; % Data seluruhnya
ys = A(2:2408,6:7)'; % Data seluruhnya

[rowv,colv] = size(uv);  %matrix uv
[rowu,colu] = size(ut);  %matrix ut
[rowy,coly] = size(yt);  %matrix yt
Min = -ones(rowu,1);
Max = ones(rowu,1);
MM = [Min Max];

for i=1:rowu
    maxusa(i)=max(us(i,:)); %change the range
    minusa(i)=min(us(i,:)); %perubahan maksimal us
end

for i = 1:rowy
    maxys(i)=max(ys(i,:));
    minys(i)=min(ys(i,:));
end

minmaxus = [maxusa;minusa];
minmaxys = [maxys;minys];

for i = 1:rowy
    yt(i,:)=((2/(max(ys(i,:))-min(ys(i,:))))*(yt(i,:)-
min(ys(i,:))))-1;
    yv(i,:)=((2/(max(ys(i,:))-min(ys(i,:))))*(yv(i,:)-
min(ys(i,:))))-1;
end

for j = 1:colu
    for i=1:rowu
        ut(i,j)=((2/(maxusa(i)-minusa(i)))*(ut(i,j)-minusa(i)))-1;
    end
end
```

```
for j = 1:colv
    for i=1:rowv
        uv(i,j)=((2/(maxusa(i)-minusa(i)))*(uv(i,j)-minusa(i)))-1;
    end
end

%penentuan matrix input training
ut1=ut(1,:)';
ut2=ut(2,:)';
ut3=ut(3,:)';
ut4=ut(4,:)';
ut5=ut(5,:)';

%penentuan matrix output training
yt1 = yt(1,:)';
yt2 = yt(2,:)';

%penetuan matrix input validasi
uv1=uv(1,:)';
uv2=uv(2,:)';
uv3=uv(3,:)';
uv4=uv(4,:)';
uv5=uv(5,:)';

%penentuan matrix output validasi
yv1 = yv(1,:)';
yv2 = yv(2,:)';

% history length for MIMO identification
% panjang input matrix
hist = ones(1,5);
%hist = [1 1 1 1 2 2 2 2 3 3 3 3 1 1];

[n_rows,n_col] = size(ut1);

% setting training data matrix
data_latih = zeros(n_rows-1,sum(hist));

for i = 1:hist(1),
    data_latih(:,i) = [zeros(hist(1)-i,1);ut1(2:n_rows-hist(1)+i)];
end

for j = 1:hist(2),
    data_latih(:,sum(hist(1))+j) = [zeros(hist(2)-j,1);ut2(2:n_rows-
hist(2)+j)];
end

for k = 1:hist(3),
    data_latih(:,sum(hist(1:2))+k) = [zeros(hist(3)-
k,1);ut3(2:n_rows-hist(3)+k)];
end
for l = 1:hist(4),
    data_latih(:,sum(hist(1:3))+l) = [zeros(hist(4)-
l,1);ut4(2:n_rows-hist(4)+l)];
```

```
end

for m = 1:hist(5),
    data_latih(:,sum(hist(1:4))+m) = [zeros(hist(5)-
m,1);ut5(2:n_rows-hist(5)+m)];
end

PHI = data_latih';

% Construction of output matrix
Y = zeros(n_rows-1,3);
Y(:,1) = yt1(2:end);
Y(:,2) = yt2(2:end);

Ys = Y';

%Pembentukan struktur jaringan
% Construction of networks structure
NetDef = [];
netdef1 = 'HHH'; %tanh=fungsi aktivasi
netdef2 = 'LLL';
L = [netdef1;netdef2];
Data_RMSE =[];
trparms = settrain;
x=20
for x=7 %1 = 3, 15 = 18 %Hidden Node 20
    hn = x
    close all;
    Ys = Y'
    NetDef = [NetDef L]
    netdef1 = 'HHHHHHHHHHHHH';
    netdef2 = '-------------';
    L = [netdef1;netdef2];

    % Construction of networks structure

    % NetDef = ['HHHHHHHHH';'LLL------'];
    trparms = settrain;

    %W1 = bobot awal, W2=bobot baru,  yhat=output yhat (hasil
prediksi).

    [W1,W2,PI_vec,yhat] = marq_rev(NetDef,[],[],PHI,Ys,trparms);
    % RMSE calculation
    for i = 1:2
        RMSE_train(i)=r_m_s_e(yhat(i,:),Ys(i,:))
    end

    for i = 1:2
        Ys(i,:)=(((max(ys(i,:))-
min(ys(i,:))))*(Ys(i,:)+1)/2)+min(ys(i,:));         %Descaling
        Yhat(i,:)=(((max(ys(i,:))-
min(ys(i,:))))*(yhat(i,:)+1)/2)+min(ys(i,:));          %Descaling
        RMSE_train_f(i) = r_m_s_e(Ys(i,:),Yhat(i,:));
    end
     %Drawing
```

```
for i = 1
         figure(i)
         plot(Ys(i,:),'b-');
         hold on
         plot(Yhat(i,:),'r.','LineWidth',1);
         grid
         title('Network Training ');
         legend('Solid : Actual','Dot : Predicted ',
'Location','Best');
         ylabel('Mole fraction of distillate');
         xlabel('Time (second)');
     end


     for i = 2
      figure(i)
      plot(Ys(i,:),'b-');
      hold on
      plot(Yhat(i,:),'r.','LineWidth',1);
      grid
      title('Network Training ');
      legend('Solid : Actual','Dot : Predicted ',
'Location','Best');
      ylabel('Mole fraction of bottom product');
      xlabel('Time (Second)');
     end

    save WT_Cat NetDef W1 W2 maxys minys maxusa minusa
    xlswrite('E1101', W1, 'W1')
    xlswrite('E1101', W2, 'W2')

    % Tahap Validasi
    disp('------------------------')
    disp(' VALIDATION IN PROGRESS ')
    disp('------------------------')

    [n_rows,n_col] = size(uv1);
    data_uji = zeros(n_rows-1,sum(hist));

    for i = 1:hist(1),
        data_uji(:,i) = [zeros(hist(1)-i,1);uv1(2:n_rows-
hist(1)+i)];
    end

    for j = 1:hist(2),
        data_uji(:,sum(hist(1))+j) = [zeros(hist(2)-
j,1);uv2(2:n_rows-hist(2)+j)];
    end

    for k = 1:hist(3),
        data_uji(:,sum(hist(1:2))+k) = [zeros(hist(3)-
k,1);uv3(2:n_rows-hist(3)+k)];
    end
    for l = 1:hist(4),
        data_uji(:,sum(hist(1:3))+l) = [zeros(hist(4)-
l,1);uv4(2:n_rows-hist(4)+l)];
```

```
    end

    for m = 1:hist(5),
        data_uji(:,sum(hist(1:4))+m) = [zeros(hist(5)-
m,1);uv5(2:n_rows-hist(5)+m)];
    end

    PHI_uji = data_uji';

    Y_uji = zeros(n_rows-1,3);
    Y_uji(:,1) = yv1(2:end);
    Y_uji(:,2) = yv2(2:end);

    Ys_uji = Y_uji';

    [y2_uji]=marq_rev_uji(NetDef,W1,W2,PHI_uji,Ys_uji);

    % RMSE calculation
    for i = 1:2
        RMSE_test(i)=r_m_s_e(Ys_uji(i,:),y2_uji(i,:))
    end

    for i = 1:2
        Ys_test(i,:)=(((max(ys(i,:))-
min(ys(i,:))))*(Ys_uji(i,:)+1)/2)+min(ys(i,:));    %Descaling
        Yhat_test(i,:)=(((max(ys(i,:))-
min(ys(i,:))))*(y2_uji(i,:)+1)/2)+min(ys(i,:));  %Descaling
        RMSE_test_f(i) = r_m_s_e(Ys_test(i,:),Yhat_test(i,:));
    end
    %Drawing
     for i = 1
         figure(i+2)
         plot(Ys_test(i,:),'k-');
         hold on
         plot(Yhat_test(i,:),'r.','LineWidth',1);
         grid
         title('Network Validation ');
         legend('Solid : Actual','Dot : Predicted ',
'Location','Best');
         ylabel('Mole fraction of distillate');
         xlabel('Time (Second)');
     end

       for i = 2
       figure(i+2)
       plot(Ys_test(i,:),'k-');
       hold on
       plot(Yhat_test(i,:),'r.','LineWidth',1);
       grid
       title('Network Validation ');
       legend('Solid : Actual','Dot : Predicted ',
'Location','Best');
       ylabel('Mole fraction of bottom product');
       xlabel('Time (Second)');
       end
    %================================================================
```

```
    RMSE_iterasi = [RMSE_train RMSE_test RMSE_train_f RMSE_test_f ];
    Data_RMSE = [Data_RMSE; RMSE_iterasi]

end
xlswrite('tesANN1.xls', Data_RMSE, 'data')
% save E1101_RMSEfind RMSE_train RMSE_test RMSE_train_f RMSE_test_f
```

Proses Training JST

```
function [XD, XB] = tes1(a1, a2, a3, a4, a5)%, a8, a9, a10, a11,
a12, a13)

% clear;
clc;

 a1 = 325;
 a2 = 0.98266;
 a3 = 9.64E-02;
 a4 = 2.020590
 a5 = 6121.209
load WT_Cat.mat

PHI_uji = [a1; a2; a3; a4; a5]

for i=1:5
maxusa(i)=maxusa(i); %change the range
minusa(i)=minusa(i);
end
for i=1:5
PHI_uji(i,1)=((2/(maxusa(i)-minusa(i)))*(PHI_uji(i,1)-minusa(i)))-1;
end

[yjalan]=marq_rev_jalan(NetDef,W1,W2,PHI_uji);

for i = 1:2
yjalan1(i,:)=(((maxys(i)-minys(i)))*(yjalan(i,:)+1)/2)+minys(i);
end
yhat1     = yjalan1(1,1);
yhat2     = yjalan1(2,1);

XD = yhat1
XB = yhat2

end
```

Settrain

```
function tr = settrain(trparms,varargin)
%SETTRAIN set parameters for training algorithm.
%   It is only necessary to set parameters specific to the selected
%   training algorithm. In case parameters that are needed in the
training
%   algorithm are not set, the called training function will
automatically
%   set these parameters to the default values.
```

```
%
%   TRPARMS = SETTRAIN
%      Set all parameters to default values.
%
%   SETTRAIN(TRPARMS)
%      List all parameters.
%
%   TRPARMS = SETTRAIN(TRPARMS,'field1',value1,'field2',value2,...)
%      Set specific parameters
%         TRPARMS.field1 = value1;
%         TRPARMS.filed2 = value2;
%         etc.
%      If value = 'default', the parameter is set to the default
value.
%
%   The following fields are valid:
%   Information displayed during training
%    infolevel - Display little information (0) or much (1)
%
%   Stopping criteria (all algorithms)
%    maxiter   - Maximum iterations.
%    critmin   - Stop if criterion is below this value.
%    critterm  - Stop if change in criterion is below this value.
%    gradterm  - Stop if largest element in gradient is below this
value.
%    paramterm - Stop if largest parameter change is below this
value.
%    NB: critterm, gradterm and paramterm must all be satisfied.
%
%   Weight decay (all algorithms trained with the Levenberg-
Marquardt alg.).
%    D         - Row vector containing the weight decay parameters.
If D has
%                one element a scalar weight decay will be used. If
D has two
%                elements, the first element will be used as weight
decay for
%                the hidden-to-output layer while second will be
used for the
%                input-to-hidden layer weights. For individual
weight decays,
%                D must contain as many elements as there are
weights in the
%                network.
%
%   Levenberg-Marquardt parameters
%    lambda    - Initial Levenberg-Marquardt parameter.
%
%   Backprop parameters
%    eta       - Step size.
%    alph      - Momentum.
%
%   RPE parameters
%    method    - Training method ('ff', 'ct', 'efra').
%
%    Forgetting factor
%    fflambda  - Forgetting factor.
%    p0        - Covariance matrix is initialized to p0*I.
```

```
%
%    Constant trace
%    ctlambda  - Forgetting factor.
%    alpha_min - Min. eigenvalue of P matrix.
%    alpha_max - Max. eigenvalue of P matrix.
%
%    EFRA
%    eflambda  - Forgetting factor.
%    alpha     - EFRA parameter.
%    beta      - EFRA parameter.
%    delta     - EFRA parameter.
%
%   For recurrent nets
%    skip      - Do not use the first 'skip' samples for training.
%
%   For multi-output nets
%    repeat    - Number of times the IGLS procedure should be
repeated.

%  Programmed by : Magnus Norgaard, IAU/IMM
%  LastEditDate  : Dec. 29, 1999

% >>>>>>>>>>>>>>>>>>>>>   SET ALL PARAMETERS TO DEFAULT
<<<<<<<<<<<<<<<<<<<
% Information level
trd.infolevel = 0;
rand('seed',419877);
% Termination values
trd.maxiter =150;
trd.critmin = 0;
trd.critterm = 0;
trd.gradterm = 1e-4;
trd.paramterm = 1e-3;

% Weight 3ecay
trd.D = 0;

% Levenberg-Marquardt parameters
trd.lambda = 1;

% Backprop parameters
trd.eta  = 1e-4;
trd.alph = 0;

% RPE parameters
trd.method = 'ff';
trd.fflambda = 0.995;
trd.p0 = 10;

trd.alpha_min = 1e-3;
trd.alpha_max = 1e1;

trd.eflambda = 0.995;
trd.alpha = 1;
trd.beta  = 0.001;
trd.delta = 0.001;
```

```
% For recurrent nets
trd.skip = 0;

% For multi-output nets
trd.repeat = 5;

% Default names
dnames = fieldnames(trd);

if nargin==0
  tr = trd;

% >>>>>>>>>>>>>>>>>>>>>>>>>>>   DISPLAY PROPERTIES
<<<<<<<<<<<<<<<<<<<<<<<<<<<
elseif nargin==1,
  names = fieldnames(trparms);
  for idx=1:length(names),
    tmp = getfield(trparms,names{idx});
    if ischar(tmp),
      fprintf('%15s  = %s\n',names{idx},tmp);
    elseif (size(tmp,1)==1 | size(tmp,1)==1)
      if rem(tmp,1)==0,
        fprintf('%15s  = %d\n',names{idx},tmp);
      else
        fprintf('%15s  = %4.3e\n',names{idx},tmp);
      end
    else
       fprintf('%15s  = [%dx%d
double]\n',names{idx},size(tmp,1),size(tmp,2));
    end
  end

% >>>>>>>>>>>>>>>>>>>>>>>>>>   SET SPECIFIC PROPERTIES
<<<<<<<<<<<<<<<<<<<<<<<<<<
elseif nargin>=2,
  tr = trparms;
  if rem(length(varargin),2),
    error('You must specify an even number of properties.');
  end
  for idx=1:2:length(varargin)

    % Check if field is a string
    if ~isstr(varargin{idx})
      error('Property name must be a string.');

    % Check if field is illegal

elseif(isempty(find(strcmp(lower(dnames),lower(varargin{idx})))))
       errstr = sprintf('%s ''%s''.','Unknown property
name',varargin{idx});
       error(errstr);

    % Set field to default value if requested
    elseif(strcmp(lower(varargin{idx+1}),'default'))
      if strcmp(lower(varargin{idx}),'d'),
```

```
        tr = setfield(tr,'D',getfield(trd,'D'));
      else

tr=setfield(tr,lower(varargin{idx}),getfield(trd,lower(varargin{idx}
)));
      end

    % Set field to specified value
    else
      if strcmp(lower(varargin{idx}),'d'),
        tr = setfield(tr,'D',varargin{idx+1});
      else
        tr = setfield(tr,lower(varargin{idx}),varargin{idx+1});
      end
    end
  end
end
```

RMSE

```
function [e]=r_m_s_e(y,yhat);

% function [e]=r_m_s_e(y,yhat);
%
% Fungsi ini untuk menghitung root mean squared error
% dari data hasil identifikasi
%
% y      : data dari output proses
% yhat   : data dari output model


l1=length(y);
l2=length(yhat);

if l1==l2
   e=sqrt(sum((y-yhat).^2)/l1);
else
error ('Dimensi data tidak sama')
end
```

pmntanh

```
function t=pmntanh(x)
% PMNTANH
% -------
% Fast hyperbolic tangent function to be used in
% neural networks instead of the tanh provided by MATLAB
t=1-2./(exp(2*x)+1);
```

marqrevuji

```
function [y2]=marq_rev_uji(NetDefi,W1,W2,PHI,Y)
% Fungsi untuk pelatihan jaringan syaraf tiruan
```

```
%

[outputs,N] = size(Y);                  % # of outputs and # of data
[inputs,N] = size(PHI);                 % # of hidden units
L_hidden = find(NetDefi(1,:)=='L')';     % Location of linear hidden
neurons
H_hidden = find(NetDefi(1,:)=='H')';     % Location of tanh hidden
neurons
L_output = find(NetDefi(2,:)=='L')';     % Location of linear output
neurons
H_output = find(NetDefi(2,:)=='H')';     % Location of tanh output
neurons
hidden = length(L_hidden)+length(H_hidden);
if isempty(W1) | isempty(W2),           % Initialize weights if
nescessary
  error('Networks weights is not defined')
end
if (size(W1,2)~=inputs+1 | size(W1,1)~=hidden |... % Check
dimensions
      size(W2,2)~=hidden+1 | size(W2,1)~=outputs)
   error('Dimension mismatch in weights, data, or NetDefi.');
end
y1       = [zeros(hidden,N);ones(1,N)]; % Hidden layer outputs
y2       = zeros(outputs,N);            % Network output
PHI      = [PHI;ones(1,N)];             % Augment PHI with a row
containg ones

% >>>>>>>>>>>>>>>>>>>>  COMPUTE NETWORK OUTPUT  y2(theta)
<<<<<<<<<<<<<<<<<<<<
h1 = W1*PHI;
y1(H_hidden,:) = pmntanh(h1(H_hidden,:));
y1(L_hidden,:) = h1(L_hidden,:);

h2 = W2*y1;
y2(H_output,:) = pmntanh(h2(H_output,:));
y2(L_output,:) = h2(L_output,:);
```

marq rev jalan

```
function [y2]=marq_rev_jalan(NetDefi,W1,W2,PHI)
% Fungsi untuk pelatihan jaringan syaraf tiruan
%

outputs = 2;                    % # of outputs and # of data
[inputs,N] = size(PHI);                 % # of hidden units
L_hidden = find(NetDefi(1,:)=='L')';     % Location of linear hidden
neurons
H_hidden = find(NetDefi(1,:)=='H')';     % Location of tanh hidden
neurons
L_output = find(NetDefi(2,:)=='L')';     % Location of linear output
neurons
H_output = find(NetDefi(2,:)=='H')';     % Location of tanh output
neurons
hidden = length(L_hidden)+length(H_hidden);
```

```
if isempty(W1) | isempty(W2),          % Initialize weights if
nescessary
  error('Networks weights is not defined')
end
if (size(W1,2)~=inputs+1 | size(W1,1)~=hidden |... % Check
dimensions
      size(W2,2)~=hidden+1 | size(W2,1)~=outputs)
   error('Dimension mismatch in weights, data, or NetDefi.');
end
y1       = [zeros(hidden,N);ones(1,N)]; % Hidden layer outputs
y2       = zeros(outputs,N);            % Network output
PHI      = [PHI;ones(1,N)];             % Augment PHI with a row
containg ones

% >>>>>>>>>>>>>>>>>>>>  COMPUTE NETWORK OUTPUT  y2(theta)
<<<<<<<<<<<<<<<<<<<<
h1 = W1*PHI;
y1(H_hidden,:) = pmntanh(h1(H_hidden,:));
y1(L_hidden,:) = h1(L_hidden,:);

h2 = W2*y1;
y2(H_output,:) = pmntanh(h2(H_output,:));
y2(L_output,:) = h2(L_output,:);
```

marq rev

```
function [W1,W2,PI_vector,y2]=marq_rev(NetDefi,W1,W2,PHI,Y,trparms)
% Fungsi untuk pelatihan jaringan syaraf tiruan
%
[outputs,N] = size(Y);                 % # of outputs and # of data
[inputs,N] = size(PHI);                % # of hidden units
L_hidden = find(NetDefi(1,:)=='L')';     % Location of linear hidden
neurons
H_hidden = find(NetDefi(1,:)=='H')';     % Location of tanh hidden
neurons
L_output = find(NetDefi(2,:)=='L')';     % Location of linear output
neurons
H_output = find(NetDefi(2,:)=='H')';     % Location of tanh output
neurons
hidden = length(L_hidden)+length(H_hidden);
if isempty(W1) | isempty(W2),          % Initialize weights if
nescessary
  W1 = rand(hidden,inputs+1)-0.5;
  W2 = rand(outputs,hidden+1)-0.5;
end
if (size(W1,2)~=inputs+1 | size(W1,1)~=hidden |... % Check
dimensions
      size(W2,2)~=hidden+1 | size(W2,1)~=outputs)
   error('Dimension mismatch in weights, data, or NetDefi.');
end
y1       = [zeros(hidden,N);ones(1,N)]; % Hidden layer outputs
y2       = zeros(outputs,N);            % Network output
index = outputs*(hidden+1) + 1 + [0:hidden-1]*(inputs+1); % A useful
vector!
index2 = (0:N-1)*outputs;              % Yet another useful vector
```

```
iteration = 1;                          % Counter variable
dw        = 1;                          % Flag telling that the
weights are new
PHI       = [PHI;ones(1,N)];            % Augment PHI with a row
containg ones
parameters1= hidden*(inputs+1);         % # of input-to-hidden
weights
parameters2= outputs*(hidden+1);        % # of hidden-to-output
weights
parameters = parameters1 + parameters2; % Total # of weights
PSI       = zeros(parameters,outputs*N); % Deriv. of each output
w.r.t. each weight
ones_h    = ones(hidden+1,1);           % A vector of ones
ones_i    = ones(inputs+1,1);           % Another vector of ones
                                        % Parameter vector
containing all weights
theta = [reshape(W2',parameters2,1) ; reshape(W1',parameters1,1)];
theta_index = find(theta);              % Index to weights<>0
theta_red = theta(theta_index);         % Reduced parameter vector
reduced  = length(theta_index);         % The # of parameters in
theta_red
index3   = 1:(reduced+1):(reduced^2);   % A third useful vector
lambda_old = 0;
if nargin<6 | isempty(trparms) % Default training parameters
  trparms = settrain;
  lambda  = trparms.lambda;
  D       = trparms.D;
else                                    % User specified values
  if ~isstruct(trparms),
     error('''trparms'' must be a structure variable.');
  end
  if ~isfield(trparms,'infolevel')
     trparms = settrain(trparms,'infolevel','default');
  end
  if ~isfield(trparms,'maxiter')
     trparms = settrain(trparms,'maxiter','default');
  end
  if ~isfield(trparms,'critmin')
     trparms = settrain(trparms,'critmin','default');
  end
  if ~isfield(trparms,'critterm')
     trparms = settrain(trparms,'critterm','default');
  end
  if ~isfield(trparms,'gradterm')
     trparms = settrain(trparms,'gradterm','default');
  end
  if ~isfield(trparms,'paramterm')
     trparms = settrain(trparms,'paramterm','default');
  end
  if ~isfield(trparms,'lambda')
     trparms = settrain(trparms,'lambda','default');
  end
  lambda    = trparms.lambda;
  if ~isfield(trparms,'D')
     trparms = settrain(trparms,'D','default');
     D = trparms.D;
  else
```

```
    if length(trparms.D)==1,              % Scalar weight decay
parameter
      D = trparms.D(ones(1,reduced));
    elseif length(trparms.D)==2,          % Two weight decay
parameters
      D = trparms.D([ones(1,parameters2) 2*ones(1,parameters1)])';
      D = D(theta_index);
    elseif length(trparms.D)>2,           % Individual weight decay
      D = trparms.D(:);
    end
  end
end
D = D(:);
critdif  = trparms.critterm+1;            % Initialize stopping
variables
gradmax  = trparms.gradterm+1;
paramdif = trparms.paramterm+1;
PI_vector = zeros(trparms.maxiter,1);     % Vector for storing
criterion values


%-----------------------------------------------------------------------
---------------
%--------------                    TRAIN NETWORK
-------------
%-----------------------------------------------------------------------
---------------
clc;
c=fix(clock);
fprintf('Network training started at
%2i.%2i.%2i\n\n',c(4),c(5),c(6));


% >>>>>>>>>>>>>>>>>>>>  COMPUTE NETWORK OUTPUT  y2(theta)
<<<<<<<<<<<<<<<<<<<<
h1 = W1*PHI;
y1(H_hidden,:) = pmntanh(h1(H_hidden,:));
y1(L_hidden,:) = h1(L_hidden,:);

h2 = W2*y1;
y2(H_output,:) = pmntanh(h2(H_output,:));
y2(L_output,:) = h2(L_output,:);

E        = Y - y2;                        % Training error
E_vector = E(:);                          % Reshape E into a long
vector
SSE      = E_vector'*E_vector;            % Sum of squared errors
(SSE)
PI       = (SSE+theta_red'*(D.*theta_red))/(2*N); % Performance
index

% Iterate until stopping criterion is satisfied
while (iteration<=trparms.maxiter & PI>trparms.critmin & lambda<1e7
& ...
       (critdif>trparms.critterm | gradmax>trparms.gradterm | ...
       paramdif>trparms.paramterm))
```

```
    if dw==1,
   % >>>>>>>>>>>>>>>>>>>>>>>>>   COMPUTE THE PSI MATRIX
<<<<<<<<<<<<<<<<<<<<<<<<<
   % (The derivative of each network output (y2) with respect to
each weight)

    % ==========   Elements corresponding to the linear output units
============
    for i = L_output'
      index1 = (i-1) * (hidden + 1) + 1;

      % -- The part of PSI corresponding to hidden-to-output layer
weights --
      PSI(index1:index1+hidden,index2+i) = y1;
      % ----------------------------------------------------------
---------

      % -- The part of PSI corresponding to input-to-hidden layer
weights ---
      for j = L_hidden',
        PSI(index(j):index(j)+inputs,index2+i) = W2(i,j)*PHI;
      end

      for j = H_hidden',
        tmp = W2(i,j)*(1-y1(j,:).*y1(j,:));
        PSI(index(j):index(j)+inputs,index2+i) = tmp(ones_i,:).*PHI;
      end
      % ----------------------------------------------------------
---------
    end

    % ============  Elements corresponding to the tanh output units
==============
    for i = H_output',
      index1 = (i-1) * (hidden + 1) + 1;

      % -- The part of PSI corresponding to hidden-to-output layer
weights --
      tmp = 1 - y2(i,:).*y2(i,:);
      PSI(index1:index1+hidden,index2+i) = y1.*tmp(ones_h,:);
      % ----------------------------------------------------------
---------

      % -- The part of PSI corresponding to input-to-hidden layer
weights ---
      for j = L_hidden',
        tmp = W2(i,j)*(1-y2(i,:).*y2(i,:));
        PSI(index(j):index(j)+inputs,index2+i) = tmp(ones_i,:).*PHI;
      end

      for j = H_hidden',
        tmp  = W2(i,j)*(1-y1(j,:).*y1(j,:));
        tmp2 = (1-y2(i,:).*y2(i,:));
        PSI(index(j):index(j)+inputs,index2+i) = tmp(ones_i,:)...
```

```
.*tmp2(ones_i,:).*PHI;
      end
      % ------------------------------------------------------------
---------
    end
    PSI_red = PSI(theta_index,:);

    % -- Gradient --
    G = PSI_red*E_vector-D.*theta_red;

    % -- Means square error part Hessian  --
    H = PSI_red*PSI_red';
    H(index3) = H(index3)'+D;                       % Add diagonal
matrix
    dw = 0;
  end


% >>>>>>>>>>>>>>>>>>>>>>>>>>>        COMPUTE h_k
<<<<<<<<<<<<<<<<<<<<<<<<<<
  % -- Hessian  --
  H(index3) = H(index3)'+(lambda-lambda_old);       % Add diagonal
matrix

  % -- Search direction --
  h = H\G;                                          % Solve for
search direction

  % -- Compute 'apriori' iterate --
  theta_red_new = theta_red + h;                    % Update
parameter vector
  theta(theta_index) = theta_red_new;

  % -- Put the parameters back into the weight matrices --
  W1_new =
reshape(theta(parameters2+1:parameters),inputs+1,hidden)';
  W2_new = reshape(theta(1:parameters2),hidden+1,outputs)';


% >>>>>>>>>>>>>>>>>>>>   COMPUTE NETWORK OUTPUT  y2(theta+h)
<<<<<<<<<<<<<<<<<<<<
  h1 = W1_new*PHI;
  y1(H_hidden,:) = pmntanh(h1(H_hidden,:));
  y1(L_hidden,:) = h1(L_hidden,:);

  h2 = W2_new*y1;
  y2(H_output,:) = pmntanh(h2(H_output,:));
  y2(L_output,:) = h2(L_output,:);

  E_new        = Y - y2;                % Training error
  E_new_vector = E_new(:);              % Reshape E into a long
vector
  SSE_new  = E_new_vector'*E_new_vector; % Sum of squared errors
(SSE)
```

```
  PI_new   = (SSE_new + theta_red_new'*(D.*theta_red_new))/(2*N); %
PI


% >>>>>>>>>>>>>>>>>>>>>>>>>>>       UPDATE  lambda
<<<<<<<<<<<<<<<<<<<<<<<<<<<
  L = h'*G + h'*(h.*(D+lambda));
  lambda_old = lambda;

  % Decrease lambda if SSE has fallen 'sufficiently'
  if 2*N*(PI - PI_new) > (0.75*L),
    lambda = lambda/2;

  % Increase lambda if SSE has grown 'sufficiently'
  elseif 2*N*(PI-PI_new) <= (0.25*L),
    lambda = 2*lambda;
  end


% >>>>>>>>>>>>>>>>>>>       UPDATES FOR NEXT ITERATION
<<<<<<<<<<<<<<<<<<<
  % Update only if criterion has decreased
  if PI_new < PI,
    critdif  = PI-PI_new;                          % Criterion
difference
    gradmax  = max(abs(G))/N;                      % Maximum
gradient
    paramdif = max(abs(theta_red_new - theta_red)); % Maximum
parameter dif.
    W1 = W1_new;
    W2 = W2_new;
    theta_red = theta_red_new;
    E_vector = E_new_vector;
    PI = PI_new;
    dw = 1;
    lambda_old = 0;
    iteration = iteration + 1;
    PI_vector(iteration-1) = PI;                                %
Collect PI in vector
    switch(trparms.infolevel)                                   % Print
on-line inform
       case 1
          fprintf('# %i   W=%4.3e  critdif=%3.2e  maxgrad=%3.2e
paramdif=%3.2e\n',...
                                                    iteration-
1,PI,critdif,gradmax,paramdif);
       otherwise
          fprintf('iteration # %i   W = %4.3e\r',iteration-1,PI);
    end
  end
end
%-------------------------------------------------------------------
---------------
%--------------              END OF NETWORK TRAINING
-------------
```

```
%-------------------------------------------------------------------------
---------------
iteration = iteration-1;
PI_vector = PI_vector(1:iteration);
c=fix(clock);
fprintf('\n\nNetwork training ended at
%2i.%2i.%2i\n',c(4),c(5),c(6));
```

LAMPIRAN E

ImperialistCompetitveAlgorithm_GlobalOptimizationStrategy

```
%% Imperialist Competitive Algorithm (CCA);
% A Socio Politically Inspired Optimization Strategy.
% 2008
% To use this code, you should only prepare your cost function and
apply
% CCA to it. Please read the guide available in my home page.
% Special thank is for my friend Mostapha Kalami Heris whos breadth
wision toward
% artificial intelligence and his programming skils have always been
a
% sourse of inspiration for me. He helped me a lot to prepare this
code.
% His email: sm.kalami@gmail.com
% ---------------------------------------
% Esmaeil Atashpaz Gargari
% Control and Intelligent Processing Center of Excellence,
% ECE school of University of Tehran, Iran
% Cellphone: (+98)-932-9011620
% Email: e.atashpaz@ece.ut.ac.ir & atashpaz.gargari@gmail.com
% Home Page: http://www.atashpaz.com

close all
clc; clear
%% Problem Statement
ProblemParams.CostFuncName = 'BenchmarkFunction';    % You should
state the name of your cost function here.
ProblemParams.CostFuncExtraParams = 1;
ProblemParams.NPar = 5;                              % Number of
optimization variables of your objective function. "NPar" is the
dimention of the optimization problem.
ProblemParams.VarMin = 0;                            % Lower limit of
the optimization parameters. You can state the limit in two ways. 1)
2)
ProblemParams.VarMax = 100;                          % Lower limit of
the optimization parameters. You can state the limit in two ways. 1)
2)

% Modifying the size of VarMin and VarMax to have a general form
if numel(ProblemParams.VarMin)==1

ProblemParams.VarMin=repmat(ProblemParams.VarMin,1,ProblemParams.NPa
r);

ProblemParams.VarMax=repmat(ProblemParams.VarMax,1,ProblemParams.NPa
r);
end

ProblemParams.SearchSpaceSize = ProblemParams.VarMax -
ProblemParams.VarMin;

%% Algorithmic Parameter Setting
```

```
AlgorithmParams.NumOfCountries = 200;               % Number of
initial countries.
AlgorithmParams.NumOfInitialImperialists = 8;      % Number of
Initial Imperialists.
AlgorithmParams.NumOfAllColonies = AlgorithmParams.NumOfCountries -
AlgorithmParams.NumOfInitialImperialists;
AlgorithmParams.NumOfDecades = 500;
AlgorithmParams.RevolutionRate = 0.3;               % Revolution is
the process in which the socio-political characteristics of a
country change suddenly.
AlgorithmParams.AssimilationCoefficient = 2;        % In the
original paper assimilation coefficient is shown by "beta".
AlgorithmParams.AssimilationAngleCoefficient = .5;  % In the
original paper assimilation angle coefficient is shown by "gama".
AlgorithmParams.Zeta = 0.02;                        % Total Cost of
Empire = Cost of Imperialist + Zeta * mean(Cost of All Colonies);
AlgorithmParams.DampRatio = 0.99;
AlgorithmParams.StopIfJustOneEmpire = false;         % Use "true" to
stop the algorithm when just one empire is remaining. Use "false" to
continue the algorithm.
AlgorithmParams.UnitingThreshold = 0.02;            % The percent of
Search Space Size, which enables the uniting process of two Empires.

zarib = 1.05;                       % **** Zarib is used to prevent
the weakest impire to have a probability equal to zero
alpha = 0.1;                        % **** alpha is a number in the
interval of [0 1] but alpha<<1. alpha denotes the importance of mean
minimum compare to the global mimimum.

%% Display Setting
DisplayParams.PlotEmpires = false;    % "true" to plot. "false" to
cancel ploting.
if DisplayParams.PlotEmpires
    DisplayParams.EmpiresFigureHandle = figure('Name','Plot of
Empires','NumberTitle','off');
    DisplayParams.EmpiresAxisHandle = axes;
end

DisplayParams.PlotCost = true;    % "true" to plot. "false"
if DisplayParams.PlotCost
    DisplayParams.CostFigureHandle = figure('Name','Plot of Minimum
and Mean Costs','NumberTitle','off');
    DisplayParams.CostAxisHandle = axes;
end

ColorMatrix = [1   0   0  ; 0 1   0    ; 0   0 1    ; 1   1   0  ; 1
0 1    ; 0 1   1    ; 1 1 1       ;
               0.5 0.5 0.5; 0 0.5 0.5  ; 0.5 0 0.5  ; 0.5 0.5 0  ;
0.5 0 0    ; 0 0.5 0    ; 0 0 0.5     ;
               1   0.5 1  ; 0.1*[1 1 1]; 0.2*[1 1 1]; 0.3*[1 1 1];
0.4*[1 1 1]; 0.5*[1 1 1]; 0.6*[1 1 1]];
DisplayParams.ColorMatrix = [ColorMatrix ; sqrt(ColorMatrix)];

DisplayParams.AxisMargin.Min = ProblemParams.VarMin;
DisplayParams.AxisMargin.Max = ProblemParams.VarMax;
```

```
%% Creation of Initial Empires
InitialCountries = GenerateNewCountry(AlgorithmParams.NumOfCountries
, ProblemParams);

% Calculates the cost of each country. The less the cost is, the
more is the power.
if isempty(ProblemParams.CostFuncExtraParams)
    InitialCost =
feval(ProblemParams.CostFuncName,InitialCountries);
else
    InitialCost =
feval(ProblemParams.CostFuncName,InitialCountries,ProblemParams.Cost
FuncExtraParams);
end
[InitialCost,SortInd] = sort(InitialCost);
% Sort the cost in assending order. The best countries will be in
higher places
InitialCountries = InitialCountries(SortInd,:);
% Sort the population with respect to their cost.

Empires =
CreateInitialEmpires(InitialCountries,InitialCost,AlgorithmParams,
ProblemParams);

%% Main Loop
MinimumCost = repmat(nan,AlgorithmParams.NumOfDecades,1);
MeanCost = repmat(nan,AlgorithmParams.NumOfDecades,1);

if DisplayParams.PlotCost
    axes(DisplayParams.CostAxisHandle);
    if any(findall(0)==DisplayParams.CostFigureHandle)

h_MinCostPlot=plot(MinimumCost,'r','LineWidth',1.5,'YDataSource','Mi
nimumCost');
        hold on;

h_MeanCostPlot=plot(MeanCost,'k:','LineWidth',1.5,'YDataSource','Mea
nCost');
        hold off;
        pause(0.05);
    end
end

for Decade = 1:AlgorithmParams.NumOfDecades
    AlgorithmParams.RevolutionRate = AlgorithmParams.DampRatio *
AlgorithmParams.RevolutionRate;
    clc;
    Remained = AlgorithmParams.NumOfDecades - Decade
    for ii = 1:numel(Empires)
        %% Assimilation;  Movement of Colonies Toward Imperialists
(Assimilation Policy)
        Empires(ii) =
AssimilateColonies(Empires(ii),AlgorithmParams,ProblemParams);

        %% Revolution;  A Sudden Change in the Socio-Political
Characteristics
```

```
        Empires(ii) =
RevolveColonies(Empires(ii),AlgorithmParams,ProblemParams);

        %% New Cost Evaluation
        if isempty(ProblemParams.CostFuncExtraParams)
            Empires(ii).ColoniesCost =
feval(ProblemParams.CostFuncName,Empires(ii).ColoniesPosition);
        else
            Empires(ii).ColoniesCost =
feval(ProblemParams.CostFuncName,Empires(ii).ColoniesPosition,Proble
mParams.CostFuncExtraParams);
        end

        %% Empire Possession  (****** Power Possession, Empire
Possession)
        Empires(ii) = PossesEmpire(Empires(ii));

        %% Computation of Total Cost for Empires
        Empires(ii).TotalCost = Empires(ii).ImperialistCost +
AlgorithmParams.Zeta * mean(Empires(ii).ColoniesCost);

    end

    %% Uniting Similiar Empires
    Empires =
UniteSimilarEmpires(Empires,AlgorithmParams,ProblemParams);

    %% Imperialistic Competition
    Empires = ImperialisticCompetition(Empires);

    if numel(Empires) == 1 && AlgorithmParams.StopIfJustOneEmpire
        break
    end

    %% Displaying the Results

DisplayEmpires(Empires,AlgorithmParams,ProblemParams,DisplayParams);

    ImerialistCosts = [Empires.ImperialistCost];
    MinimumCost(Decade) = min(ImerialistCosts);
    MeanCost(Decade) = mean(ImerialistCosts);

    if DisplayParams.PlotCost
        refreshdata(h_MinCostPlot);
        refreshdata(h_MeanCostPlot);
        drawnow;
        pause(0.01);
    end
        for i = 1
            figure(i)
            %title('fitness value');
            legend('Solid : Minimum Cost','Dot : Mean Cost',
'Location','Best');
            ylabel('Cost Function');
            xlabel('Revolution');
       end
```

```
end % End of Algorithm
MinimumCost(end)
```

UniteSimilarEmpires

```
function
Empires=UniteSimilarEmpires(Empires,AlgorithmParams,ProblemParams)

    ThereshholdDistance = AlgorithmParams.UnitingThreshold *
norm(ProblemParams.SearchSpaceSize);
    NumOfEmpires = numel(Empires);

    for ii = 1:NumOfEmpires-1
        for jj = ii+1:NumOfEmpires
            DistanceVector = Empires(ii).ImperialistPosition -
Empires(jj).ImperialistPosition;
            Distance = norm(DistanceVector);
            if Distance<=ThereshholdDistance
                if Empires(ii).ImperialistCost <
Empires(jj).ImperialistCost
                    BetterEmpireInd=ii;
                    WorseEmpireInd=jj;
                else
                    BetterEmpireInd=jj;
                    WorseEmpireInd=ii;
                end

                Empires(BetterEmpireInd).ColoniesPosition =
[Empires(BetterEmpireInd).ColoniesPosition

Empires(WorseEmpireInd).ImperialistPosition

Empires(WorseEmpireInd).ColoniesPosition];

                Empires(BetterEmpireInd).ColoniesCost =
[Empires(BetterEmpireInd).ColoniesCost

Empires(WorseEmpireInd).ImperialistCost

Empires(WorseEmpireInd).ColoniesCost];

                % Update TotalCost for new United Empire
                Empires(BetterEmpireInd).TotalCost =
Empires(BetterEmpireInd).ImperialistCost + AlgorithmParams.Zeta *
mean(Empires(BetterEmpireInd).ColoniesCost);

                Empires = Empires([1:WorseEmpireInd-1
WorseEmpireInd+1:end]);
                return;
```

```
            end

        end
    end

end
```

Test1

```
function [objtv] = tes1(x)%, a8, a9, a10, a11, a12, a13)
% clear;
% clc;
size(x,1)
for iter = 1:size(x,1)

     b1= 325;
     b2=0.98266;
     b3= 9.64E-02;
     b4 = ((x(iter,4)/100) * 99.386532) +2.020590
     b5 = ((x(iter,5)/100) * 90000000) + 6121.209

    load WT_Cat.mat

    PHI_uji = [b1; b2; b3; b4; b5];

    for i=1:5
        maxusa(i)=maxusa(i); %change the range
        minusa(i)=minusa(i);
    end

    for i=1:5
        PHI_uji(i,1)=((2/(maxusa(i)-minusa(i)))*(PHI_uji(i,1)-
minusa(i)))-1;
    end

    [yjalan]=marq_rev_jalan(NetDef,W1,W2,PHI_uji);

    for i = 1:2;
        yjalan1(i,:)=(((maxys(i)-
minys(i)))*(yjalan(i,:)+1)/2)+minys(i);     %Descaling
        %Yhat(i,:)=(((max(ys(i,:))-
min(ys(i,:))))*(yhat(i,:)+1)/2)+min(ys(i,:));   %Descaling
        %RMSE_train_f(i) = r_m_s_e(Ys(i,:),Yhat(i,:));
    end

    yhat1= yjalan1(1,1);
    yhat2= yjalan1(2,1);
    XDsp = 0.99
    XBsp = 0.01
    XD(iter) = abs(XDsp-yhat1);
    XB(iter) = abs(XBsp-yhat2);

    % Objective function (minimize) if (maximize) then multiply by 1
    objtvtemp(iter) =1/((XB(iter)*10)+XD(iter));
```

```
    objtv = objtvtemp;
end


end
```

RevolveColonies

```
function TheEmpire =
RevolveColonies(TheEmpire,AlgorithmParams,ProblemParams)
    NumOfRevolvingColonies = round(AlgorithmParams.RevolutionRate *
numel(TheEmpire.ColoniesCost));
    RevolvedPosition = GenerateNewCountry(NumOfRevolvingColonies ,
ProblemParams);
    R = randperm(numel(TheEmpire.ColoniesCost));
    R = R(1:NumOfRevolvingColonies);
    TheEmpire.ColoniesPosition(R,:) = RevolvedPosition;
end
```

PossesEmpire

```
function TheEmpire = PossesEmpire(TheEmpire)

    ColoniesCost = TheEmpire.ColoniesCost;

    [MinColoniesCost BestColonyInd]=min(ColoniesCost);
    if MinColoniesCost < TheEmpire.ImperialistCost

        OldImperialistPosition = TheEmpire.ImperialistPosition;
        OldImperialistCost = TheEmpire.ImperialistCost;

        TheEmpire.ImperialistPosition =
TheEmpire.ColoniesPosition(BestColonyInd,:);
        TheEmpire.ImperialistCost =
TheEmpire.ColoniesCost(BestColonyInd);

        TheEmpire.ColoniesPosition(BestColonyInd,:) =
OldImperialistPosition;
        TheEmpire.ColoniesCost(BestColonyInd) = OldImperialistCost;
    end

end
```

GenerateNewCountry

```
function NewCountry =
GenerateNewCountry(NumOfCountries,ProblemParams)

    VarMinMatrix = repmat(ProblemParams.VarMin,NumOfCountries,1);
    VarMaxMatrix = repmat(ProblemParams.VarMax,NumOfCountries,1);
    NewCountry = (VarMaxMatrix - VarMinMatrix) .*
rand(size(VarMinMatrix)) + VarMinMatrix;
```

```
end
```

DisplayEmpires

```
function
DisplayEmpires(Empires,AlgorithmParams,ProblemParams,DisplayParams)

    if ~DisplayParams.PlotEmpires
        return;
    end

    if (ProblemParams.NPar ~= 2) && (ProblemParams.NPar ~= 3)
        return;
    end

    if ~any(findall(0)==DisplayParams.EmpiresFigureHandle)
        return;
    end


    if ProblemParams.NPar == 2
        for ii = 1:numel(Empires)

plot(DisplayParams.EmpiresAxisHandle,Empires(ii).ImperialistPosition
(1),Empires(ii).ImperialistPosition(2),'p',...
                'MarkerEdgeColor','k',...

'MarkerFaceColor',DisplayParams.ColorMatrix(ii,:),...

'MarkerSize',70*numel(Empires(ii).ColoniesCost)/AlgorithmParams.NumO
fAllColonies + 13);
            hold on


plot(DisplayParams.EmpiresAxisHandle,Empires(ii).ColoniesPosition(:,
1),Empires(ii).ColoniesPosition(:,2),'ok',...
                'MarkerEdgeColor','k',...

'MarkerFaceColor',DisplayParams.ColorMatrix(ii,:),...
                'MarkerSize',8);
        end

        xlim([DisplayParams.AxisMargin.Min(1)
DisplayParams.AxisMargin.Max(1)]);
        ylim([DisplayParams.AxisMargin.Min(2)
DisplayParams.AxisMargin.Max(2)]);
        hold off
    end

    if  ProblemParams.NPar == 3
        figure(1)
        for ii = 1:numel(Empires)

plot3(DisplayParams.EmpiresAxisHandle,Empires(ii).ImperialistPositio
n(1),Empires(ii).ImperialistPosition(2),Empires(ii).ImperialistPosit
ion(3),'p',...
                'MarkerEdgeColor','k',...
```

```
'MarkerFaceColor',DisplayParams.ColorMatrix(ii,:),...

'MarkerSize',70*numel(Empires(ii).ColoniesCost)/AlgorithmParams.NumO
fAllColonies + 13);
            hold on


plot3(DisplayParams.EmpiresAxisHandle,Empires(ii).ColoniesPosition(:
,1),Empires(ii).ColoniesPosition(:,2),Empires(ii).ColoniesPosition(:
,3),'ok',...
                'MarkerEdgeColor','k',...

'MarkerFaceColor',DisplayParams.ColorMatrix(ii,:),...
                'MarkerSize',8);
        end

        xlim([DisplayParams.AxisMargin.Min(1)
DisplayParams.AxisMargin.Max(1)]);
        ylim([DisplayParams.AxisMargin.Min(2)
DisplayParams.AxisMargin.Max(2)]);
        zlim([DisplayParams.AxisMargin.Min(3)
DisplayParams.AxisMargin.Max(3)]);
        hold off
    end

    pause(0.05);
end
```

CreateInitialEmpires

```
function Empires =
CreateInitialEmpires(InitialCountries,InitialCost,AlgorithmParams,
ProblemParams)

AllImperialistsPosition =
InitialCountries(1:AlgorithmParams.NumOfInitialImperialists,:);
AllImperialistsCost =
InitialCost(1:AlgorithmParams.NumOfInitialImperialists,:);

AllColoniesPosition =
InitialCountries(AlgorithmParams.NumOfInitialImperialists+1:end,:);
AllColoniesCost =
InitialCost(AlgorithmParams.NumOfInitialImperialists+1:end,:);

if max(AllImperialistsCost)>0
    AllImperialistsPower = 1.3 * max(AllImperialistsCost) -
AllImperialistsCost;
else
    AllImperialistsPower = 0.7 * max(AllImperialistsCost) -
AllImperialistsCost;
end
```

```
AllImperialistNumOfColonies =
round(AllImperialistsPower/sum(AllImperialistsPower) *
AlgorithmParams.NumOfAllColonies);
AllImperialistNumOfColonies(end) = AlgorithmParams.NumOfAllColonies
- sum(AllImperialistNumOfColonies(1:end-1));
RandomIndex = randperm(AlgorithmParams.NumOfAllColonies);

Empires(AlgorithmParams.NumOfInitialImperialists).ImperialistPositio
n = 0;

for ii = 1:AlgorithmParams.NumOfInitialImperialists
    Empires(ii).ImperialistPosition = AllImperialistsPosition(ii,:);
    Empires(ii).ImperialistCost = AllImperialistsCost(ii,:);
    R = RandomIndex(1:AllImperialistNumOfColonies(ii));
RandomIndex(AllImperialistNumOfColonies(ii)+1:end);
    Empires(ii).ColoniesPosition = AllColoniesPosition(R,:);
    Empires(ii).ColoniesCost = AllColoniesCost(R,:);
    Empires(ii).TotalCost = Empires(ii).ImperialistCost +
AlgorithmParams.Zeta * mean(Empires(ii).ColoniesCost);
end

for ii = 1:numel(Empires)
    if numel(Empires(ii).ColoniesPosition) == 0
        Empires(ii).ColoniesPosition =
GenerateNewCountry(1,ProblemParams);                %
        Empires.ColoniesCost =
feval(ProblemParams.FunctionName,Empires.ColoniesPosition);
    end
end
```

BenchmarkFunction

```
function z=BenchmarkFunction(x,number)
    if nargin<2
        error('Name or Number of function is not specified.');
    end

    switch number
        case 1
            z=tes1(x)';
        otherwise
            error('Invalid function number is used.');
    end

end
```

AssimilateColonies

```
function TheEmpire =
AssimilateColonies(TheEmpire,AlgorithmParams,ProblemParams)

% for i = 1:numel(Imperialists)
%     Imperialists{i}.Number_of_Colonies_matrix =
[Imperialists{i}.Number_of_Colonies_matrix
Imperialists{i}.Number_of_Colonies];
```

```
%
%     Imperialists_cost_matrix (i) =
Imperialists{i}.cost_just_by_itself;
%
%     Imperialists_position_matrix(i,:) = Imperialists{i}.position;

NumOfColonies = size(TheEmpire.ColoniesPosition,1);

Vector = repmat(TheEmpire.ImperialistPosition,NumOfColonies,1)-
TheEmpire.ColoniesPosition;

TheEmpire.ColoniesPosition = TheEmpire.ColoniesPosition + 2 *
AlgorithmParams.AssimilationCoefficient * rand(size(Vector)) .*
Vector;

MinVarMatrix = repmat(ProblemParams.VarMin,NumOfColonies,1);
MaxVarMatrix = repmat(ProblemParams.VarMax,NumOfColonies,1);

TheEmpire.ColoniesPosition=max(TheEmpire.ColoniesPosition,MinVarMatr
ix);
TheEmpire.ColoniesPosition=min(TheEmpire.ColoniesPosition,MaxVarMatr
ix);
```

BAB V
KESIMPULAN

5.1 Kesimpulan

Pada BAB sebelumnya analisa data serta pembahasan mengenai optimasi kolom distilasi biner dengan menggunakan ICA sehingga dapat diambil kesimpulan sebagai berikut:

1. Adanya variabel proses seperti panas reboiler, laju aliran *reflux*, laju aliran umpan, komposisi umpan dapat mempengaruhi komposisi kemurnian produk yang didapatkan.
2. Optimasi kolom distilasi biner dimodelkan dengan menggunakan jaringan syaraf tiruan dan dijalankan dengan menggunakan metode algoritma ICA dengan jumlah country sebesar 200, jumlah imperialis 8, dan jumlah revolusi sebanyak 500 decade, didapatkan hasil yang dapat mencapai global optimum dan didapatkan kondisi operasi optimal pada kolom distilasi biner sebesar 0.9891 untuk mol fraksi distilat dan 0.007 untuk mol fraksi bottom produk.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

Atashpaz-Gargari, E., & Lucas, C. (2007). Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. *2007 IEEE Congress on Evolutionary Computation, CEC 2007*, 4661–4667. http://doi.org/10.1109/CEC.2007.4425083

Biyanto, T. R., Khairansyah, M. D., Bayuaji, R., Firmanto, H., & Haksoro, T. (2015). Imperialist Competitive Algorithm (ICA) for Heat Exchanger Network (HEN) Cleaning Schedule Optimization. *Procedia Computer Science*, *72*, 5–12. http://doi.org/10.1016/j.procs.2015.12.099

Biyanto, T. R., Suganda, S. W., Matraji, Susatio, Y., Justiono, H., & Sarwono. (2015). Cleaning Schedule Optimization of Heat Exchanger Networks Using Particle Swarm Optimization, (August), 8. Retrieved from http://arxiv.org/abs/1512.0883

Caballero, J. A., & Grossmann, I. E. (2014). Optimal synthesis of thermally coupled distillation sequences using a novel MILP approach, *61*, 118–135.

Cybenko G. 1989, Approximation by superposition of A sigmoid function, Mathematic of control, signal and system, 2(4), 303-314.

Demuth, H. (2002). Neural Network Toolbox. *Networks*, *24*(1), 1–8. http://doi.org/10.1016/j.neunet.2005.10.002

Fausett, L. (2015). Fundamentals of Neural Networks. *Statewide Agricultural Land Use Baseline 2015*, *1*. http://doi.org/10.1017/CBO9781107415324.004

Gharavi, H., Ardehali, M. M., & Ghanbari-tichi, S. (2015). Imperial competitive algorithm optimization of fuzzy multi-objective design of a hybrid green power system with considerations for economics , reliability , and environmental emissions, *78*.

Hadi, S., Heydari, H., Ahmadpour, E., & Gholami, A. (2014). Journal of Natural Gas Science and Engineering Development of novel correlation for prediction of hydrate formation temperature based on intelligent optimization algorithms, *18*, 377–384.

Hosgor, E., Kucuk, T., Oksal, I. N., & Kaymak, D. B. (2014). Design and control of distillation processes for methanol – chloroform separation. *Computers and*

Chemical Engineering, *67*, 166–177. http://doi.org/10.1016/j.compchemeng.2014.03.026

Lucia, A., & McCallum, B. R. (2010). Energy targeting and minimum energy distillation column sequences. *Computers and Chemical Engineering*, *34*(6), 931–942. http://doi.org/10.1016/j.compchemeng.2009.10.006

Luyben, W. L. L. L. (1997). *Essentials of Process Control*. Mc Graw Hill.

Makarynskyy, O. (2004). Improving wave predictions with artificial neural networks. *Ocean Engineering*, *31*(5-6), 709–724. http://doi.org/10.1016/j.oceaneng.2003.05.003

Niknam, T., Fard, E. T., Ehrampoosh, S., & Rousta, A. (2011). A new hybrid imperialist competitive algorithm on data clustering, *36*(June), 293–315.

Pla-Franco, J., Lladosa, E., Loras, S., & Montón, J. B. (2015). Approach to the 1-propanol dehydration using an extractive distillation process with ethylene glycol. *Chemical Engineering and Processing: Process Intensification*, *91*, 121–129. http://doi.org/10.1016/j.cep.2015.03.007

Popova, S., Iliev, S., & Trifonov, M. (2011). Neural Network Prediction of the Electricity Consumption of Trolleybus and Tram Transport in Sofia City. *Energy, Environment and Development*, 116–120.

Ramanathan, S. P., Mukherjee, S., Dahule, R. K., Ghosh, S., Rahman, I., Tambe, S. S., … Kulkarni, B. D. (2001). Optimization of continuous distillation columns using stochastic optimization approaches. *Chemical Engineering Research and Design*, *79*(3), 310–322. http://doi.org/10.1205/026387601750281671

Samborskaya, M. A., Gusev, V. P., Gryaznova, I. A., Vdovushkina, N. S., & Volf, A. V. (2014). Crude Oil Distillation with Superheated Water Steam: Parametrical Sensitivity and Optimization. *Procedia Chemistry*, *10*, 337–342. http://doi.org/10.1016/j.proche.2014.10.057

Schaller, M., Hoffmann, K. H., Siragusa, G., Salamon, P., & Andresen, B. (2001). Numerically optimized performance of diabatic distillation columns. *Computers and Chemical Engineering*, *25*(11-12), 1537–1548. http://doi.org/10.1016/S0098-

1354(01)00717-7

Shahandeh, H., Jafari, M., Kasiri, N., & Ivakpour, J. (2015). Economic optimization of heat pump-assisted distillation columns in methanol-water separation. *Energy*, *80*, 496–508. http://doi.org/10.1016/j.energy.2014.12.006

Sorensen, E. (2014a). *Chapter 5 - Design and Operation of Batch Distillation*. *Distillation*. http://doi.org/http://dx.doi.org/10.1016/B978-0-12-386547-2.00005-3

Sorensen, E. (2014b). *Principles of Binary Distillation*. *Distillation: Fundamentals and Principles*. http://doi.org/10.1016/B978-0-12-386547-2.00004-1

Xu, S., Wang, Y., & Huang, A. (2014). Application of Imperialist Competitive Algorithm on Solving the Traveling Salesman Problem. *Algorithms*, *7*(2), 229–242. http://doi.org/10.3390/a7020229

Zaynab bozorgy, Mehdi Sadeghzade, S. avad M. (2014). Imperialist Competitive Algorithm for Improving Edge Detection. *Science, Computer Engineering, Software*, *4*(3), 227–230.

Halaman ini sengaja dikosongkan

BIODATA PENULIS

NUR FITRIYANI merupakan nama lengkap penulis. Penulis dilahirkan di kota Pemalang, Jawa Tengah pada tanggal 18 Maret 1991. Riwayat pendidikan penulis adalah MIN Bantarbolang Pemalang (1997-2003), SMP Negeri 2 Bantarbolang Pemalang (2003-2006), SMA Negeri 75 Jakarta (2006-2009). Penulis kemudian melanjutkan pendidikan S1 di Fisika-Fakultas Matematika dan Ilmu Pengetahuan Alam ITS (2009-2013). Dan kemudian melanjutkan studi S2 nya di Teknik Fisika-Fakultas Teknologi Industri ITS (2014-2016) dengan konsentrasi bidang minat rekayasa instrumentasi dan kontrol untuk menyelesaikan tugas akhirnya. Penulis dapat dihubungi melalui email: nfitrian75@gmail.com.